
175 PTAS
79
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 79

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 178507
Impreso en España-Printed in Spain-Julio 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)
Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

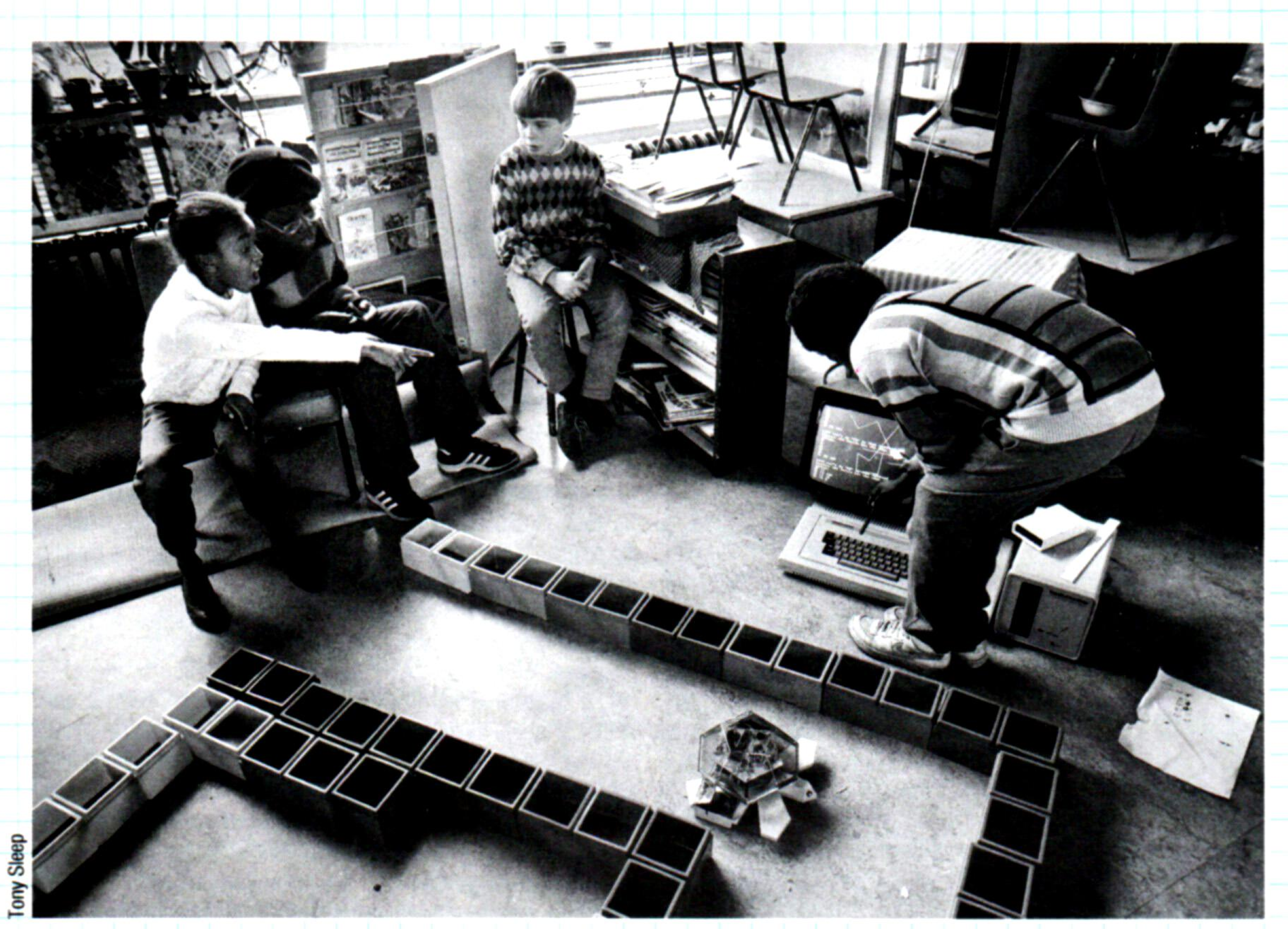
c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.




Tony Sleep

Tortuga didáctica
A través de este ejercicio con la tortuga Valiant, se está introduciendo a estos alumnos en los principios de la geometría. Esos niños toman un papel activo en la preparación del laberinto y en la programación de la tortuga para que avance evitando los obstáculos. El ejercicio enseña los principios del desplazamiento angular y otros conceptos abstractos que a los niños pequeños con frecuencia les resultan incomprensibles y nada atractivos

Poder para el alumno

Los robots, además de divertir, pueden representar un método de enseñanza nuevo y eficaz

El empleo del robot en el campo de la educación es un intento por abordar el problema (en especial, para los niños pequeños) que implica la comprensión de conceptos abstractos sin poder recurrir a una representación figurativa. Las nociones complejas de trigonometría, por ejemplo, se pueden demostrar físicamente; los procesos mentales se pueden representar mediante la realidad tridimensional de los movimientos de una tortuga. Asimismo, puede ser una útil ayuda para enseñar los principios de la programación de ordenadores.

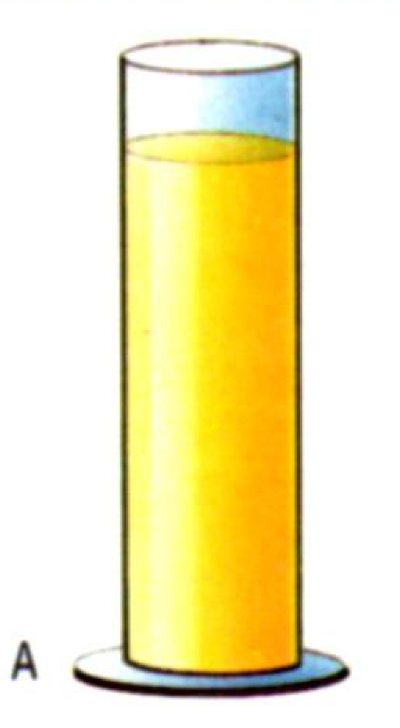
En el pasado, la geometría se ha venido enseñando con instrumentos tales como lápices, reglas y objetos similares. Un niño de ocho años de edad no suele ver mayor sentido en aprender a medir un ángulo, por no hablar de comprender los conceptos más amplios de la geometría. Los ángulos, por otra parte, adquieren muchísimo más significado cuando determinan que la tortuga, programada por el estudiante, choque o no contra el bosque de botellas de plástico colocadas sobre el suelo del aula. Aunque la geometría podría continuar resultando algo nebulosa, al menos hay una motivación para entender sus aplicaciones prácticas. La diferencia entre 15 y 20 cm se vuelve evidente enseguida tras la elección por parte de un alumno entre un ángulo de 5° y otro de 8°.

Uno de los principales aspectos de la tecnología de tortuga es que permite el aprendizaje mediante ensayo y error. El niño puede modificar la programación hasta conseguir trazar un recorrido satisfactorio, adquiriendo mientras tanto una visión constante de las propiedades de ángulos, líneas y mediciones. Un error sólo implica que se debe intentar algo más para producir los movimientos deseados, y no que se ha fallado en algo. Programar un brazo-robot para que levante y mueva un objeto exige pensar en términos de tres planos de movimiento. Imaginar lo mismo sobre el papel, que es bidimensional, no es muy fácil. Con un brazo-robot, sin embargo, los planos de movimiento resultan claramente visibles. Los resultados de la programación se observan a simple vista, los errores se distinguen fácilmente y se pueden efectuar las modificaciones pertinentes hasta conseguir el movimiento correcto. El robot le proporciona al programador una realimentación instantánea, demostrando cada ángulo y medición en secuencia con muchísima más eficacia que un ejercicio equivalente en que se utilizan los tradicionales lápiz y papel.

Un juguete robótico que en la actualidad es muy popular en las escuelas (de preescolar a primer grado) es Big Trak, un tanque futurista con una pequeña memoria a bordo y un teclado de calcula-

Ejemplo concreto

Los niños pequeños a menudo son incapaces de captar conceptos abstractos, aun cuando se les presente una demostración física de los principios implicados. Un ejemplo muy conocido es el de la conservación de cantidades y relaciones transitivas, en los que se le presentan al niño recipientes de distintas formas pero igual capacidad. En el ejemplo de abajo, A, B y C contienen la misma cantidad de líquido, y los recipientes B y B′ son idénticos. El líquido se trasvasa de un recipiente a otro y se interroga al niño

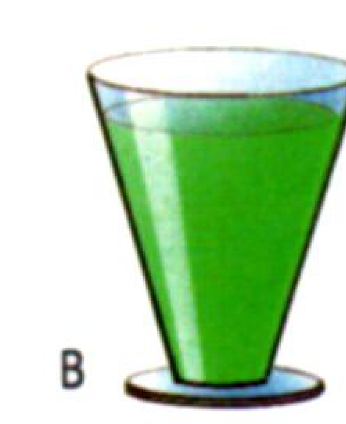


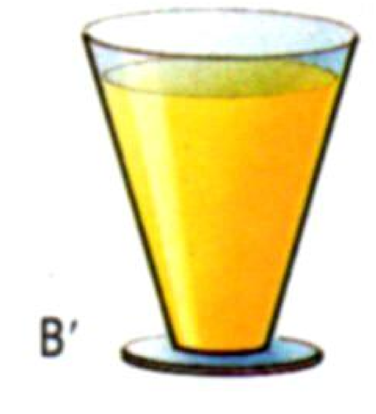


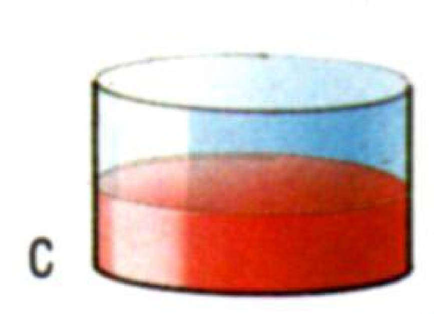


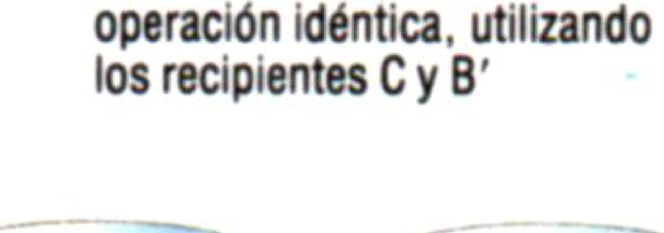


El experimentador vierte A en B′ y le pide al niño que compruebe la igualdad de los niveles de B y B′. El líquido se vuelve a verter entonces de B′ a A

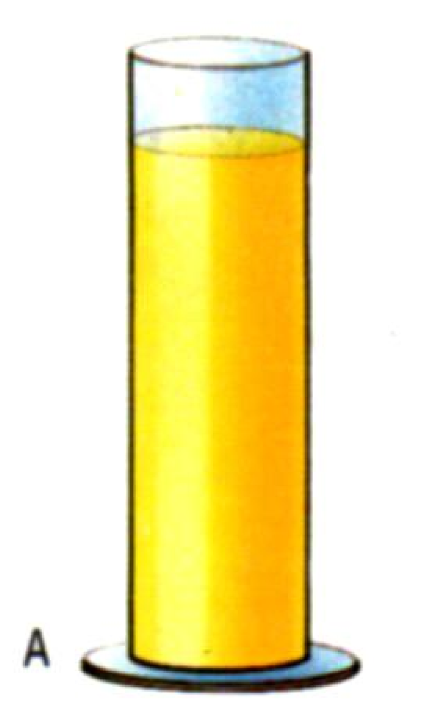


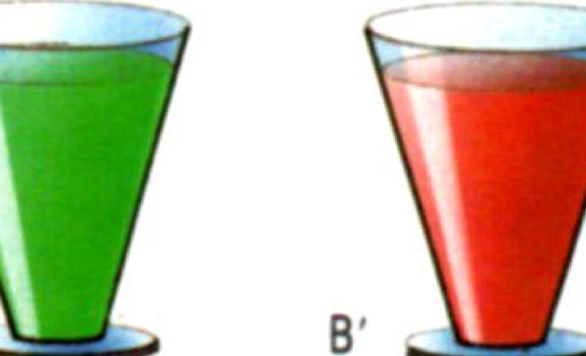


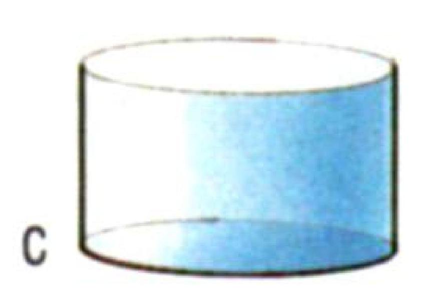




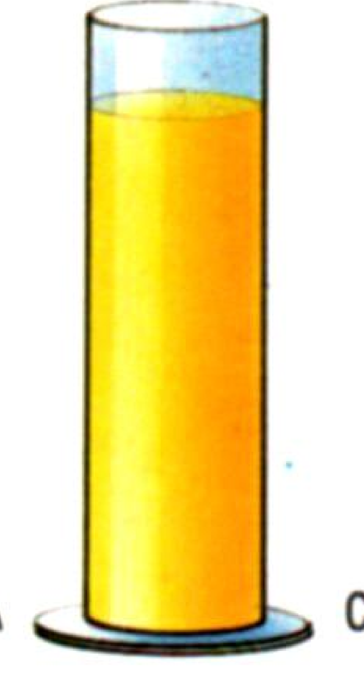
El experimentador realiza una operación idéntica, utilizando los recipientes C y B′

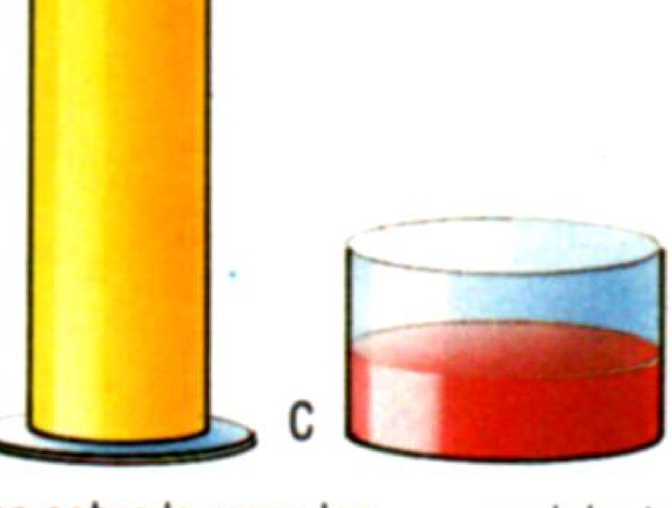


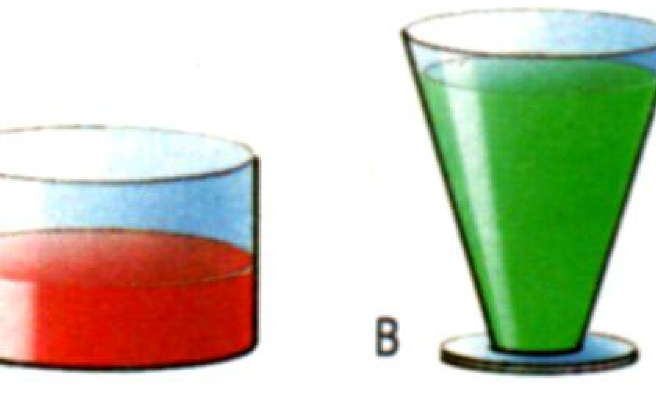


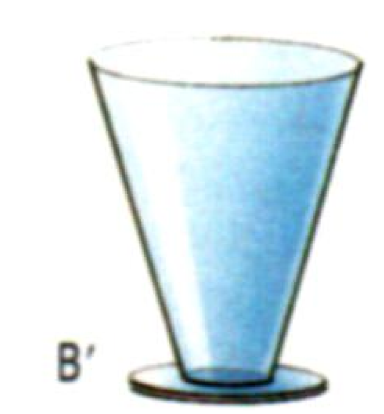


Se colocan los recipientes A y C uno junto al otro y se le pregunta al niño cuál de los dos recipientes contiene más líquido que el otro, o si ambos poseen la misma cantidad

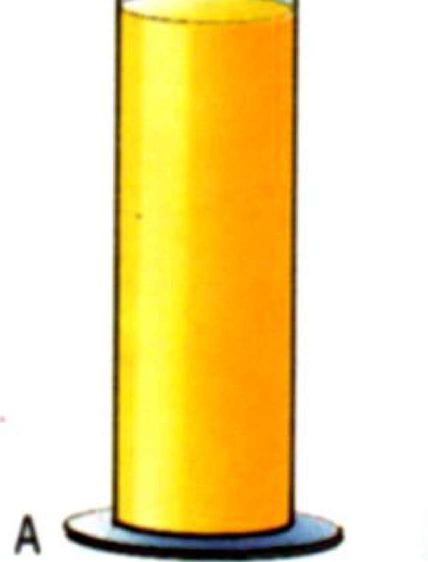








Se colocan sobre la mesa los recipientes A, B y C y se le pregunta al niño si los recipientes contienen la misma cantidad de agua. En las dos últimas etapas del experimento, si el niño ha dado una respuesta correcta, se le pide que explique su razonamiento. Por lo general los niños de menos de siete años son incapaces de captar los principios implicados o de responder acertadamente. Éste es un experimento típico de los que llevaba a cabo el psicólogo infantil Piaget, cuyo trabajo influyó en gran medida en el desarrollo del LOGO por parte de Papert. El empleo de robots en la educación introduce al niño en conceptos abstractos, al ofrecerle tanto un modelo concreto como un papel más activo en el proceso del aprendizaje







Kevin Jones

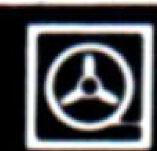
dora arriba. Teclas con flechas determinan la dirección en que se moverá el aparato, y esta instrucción ha de ir seguida por un número del 1 al 99. Forward 1 impulsará a Big Trak hacia adelante una distancia equivalente a su longitud, y Right 15 hará que efectúe un giro de 15 "minutos", o 45°, hacia la derecha. Se pueden almacenar hasta ocho instrucciones a la vez. Se lo emplea como sustituto de la tortuga, como estímulo para que el niño elabore rutinas simples de programación. El Big Trak demuestra que las instrucciones se llevan a cabo por el orden en el cual se impartieron: una iniciación muy útil a la programación de ordenadores.

Si bien las tortugas pueden ser de ayuda para la enseñanza de las matemáticas, ésta no es su función primordial; fueron diseñadas para enseñar al niño cómo programar un ordenador. El creador de la tortuga, Seymour Papert, cree que los niños deben programar el ordenador en vez de permitir que sea éste el que los programe a ellos. Esta filosofía queda patente en el lenguaje para ordenador de Papert, el LOGO, diseñado para el aprendizaje. Papert define la tortuga como "un objeto con el que pensar", en virtud del cual el niño aprende los principios básicos de la programación mediante el empleo del LOGO para controlar la tortuga.

Paul Cheung, de la Universidad de Edimburgo, desarrolló el aspecto robótico del LOGO, habiendo advertido la necesidad de un software más versátil en una época en que la mayoría de las personas estaban concentradas en el desarrollo de hardware. Cheung quería que los niños pudieran controlar diversos robots. El LOGO es ideal para programar tortugas, pero es limitado para programar otros dispositivos. Tomando el LOGO como punto de partida, Cheung reestructuró el lenguaje permitiéndole controlar más funciones y reforzando su capacidad para recibir entradas desde dispositivos tales como sensores táctiles, lectores de códigos de barras y detectores de luz. El resultado fue el Concurrent-LOGO, o C-LOGO, un lenguaje capaz de "proceso en paralelo" (la capacidad de ejecutar varias funciones simultáneamente).

El C-LOGO puede controlar hasta ocho procesos al mismo tiempo. Dos funciones particularmente interesantes son FOREVER y WHENEVER. La primera, FOREVER, iniciará un proceso y dejará que el mismo continúe ejecutándose hasta que el C-LOGO lo termine. WHENEVER espera a que aparezca una condición, momento en el que activará una función determinada; es esencial para el funcionamiento del proceso en paralelo. El C-LOGO permite activar y desactivar interruptores y también detectará si un interruptor está encendido o apagado. Esta capacidad se utiliza con el sensor: si un sensor táctil entra en contacto con una obstrucción, hace que se "encienda" un interruptor, informando al ordenador de que se ha establecido un contacto. Los niños pueden programar un *buggy* para que evite obstáculos, utilizando la función WHENEVER para ordenarle al robot que gire cada vez que se produzca un contacto.

El C-LOGO es más adecuado para los niños de escuela secundaria, donde Paul Cheung lo utilizó con diversos dispositivos de control. Niños de entre 13 y 15 años de edad han escrito programas para controlar molinos de viento, ascensores, brazos mecánicos y *buggies*, así como un programa para conectar en interface con una caja de botones que controla



los movimientos de puertas y ventanas de una casa robótica. Esta casa contenía una alarma antirrobo que se activaba al abrirse una ventana, y las puertas sólo se podían abrir tras digitar en el ordenador una contraseña. Estos dos programas se ejecutan simultáneamente mediante la incorporación del proceso en paralelo.

A las tres plantas de la casa se accedía mediante el ascensor, que estaba programado para detenerse en cada planta. No obstante, el ordenador debía saber en qué planta se hallaba el ascensor, con el fin de que el mismo funcionara correctamente. Construido con un juego de construcción Meccano, el ascensor estaba alimentado por un motor CD que tenía instalados tres interruptores de lengüeta que detectaban si el ascensor se hallaba en una planta determinada. Los niños lo programaron luego para que fuera a la planta baja pulsando un botón (de la caja de botones), otro para la segunda planta, y un tercero para la planta superior. Los niños se plantearon esta pregunta: "¿Qué sucederá si se pulsa el botón estando el ascensor en movimiento?" Su respuesta inmediata fue indicarle que no hiciera caso a ninguna señal hasta no estar detenido. Pero después de pensarlo un poco más, los niños decidieron que sería una mejor solución hacer que el ordenador llevara el registro de las instrucciones y las implementara en secuencia.

Soluciones simples

Fueron las características del C-LOGO las que les permitieron a los niños escribir de forma muy simple un programa que, de lo contrario, habría sido complejo. Se definieron como componentes esenciales un detector de señales, un secuenciador (*scheduler*) y un controlador de ascensor, actuando el detector de señales y el detector del ascensor como procesos paralelos. Las señales se detectaban utilizando FOREVER y se le pasaban luego al secuenciador. El controlador del ascensor le enviaba entonces un mensaje al secuenciador pidiéndole un destino, con lo que el ascensor se trasladaba a la planta apropiada y después solicitaba otro destino. Tras almacenar las instrucciones en secuencia, el secuenciador las enviaba por orden al controlador. Entusiasmados por tener el ascensor a sus órdenes, así como por contar con los medios para controlar su programación, los niños perseveraron hasta resolver el problema.

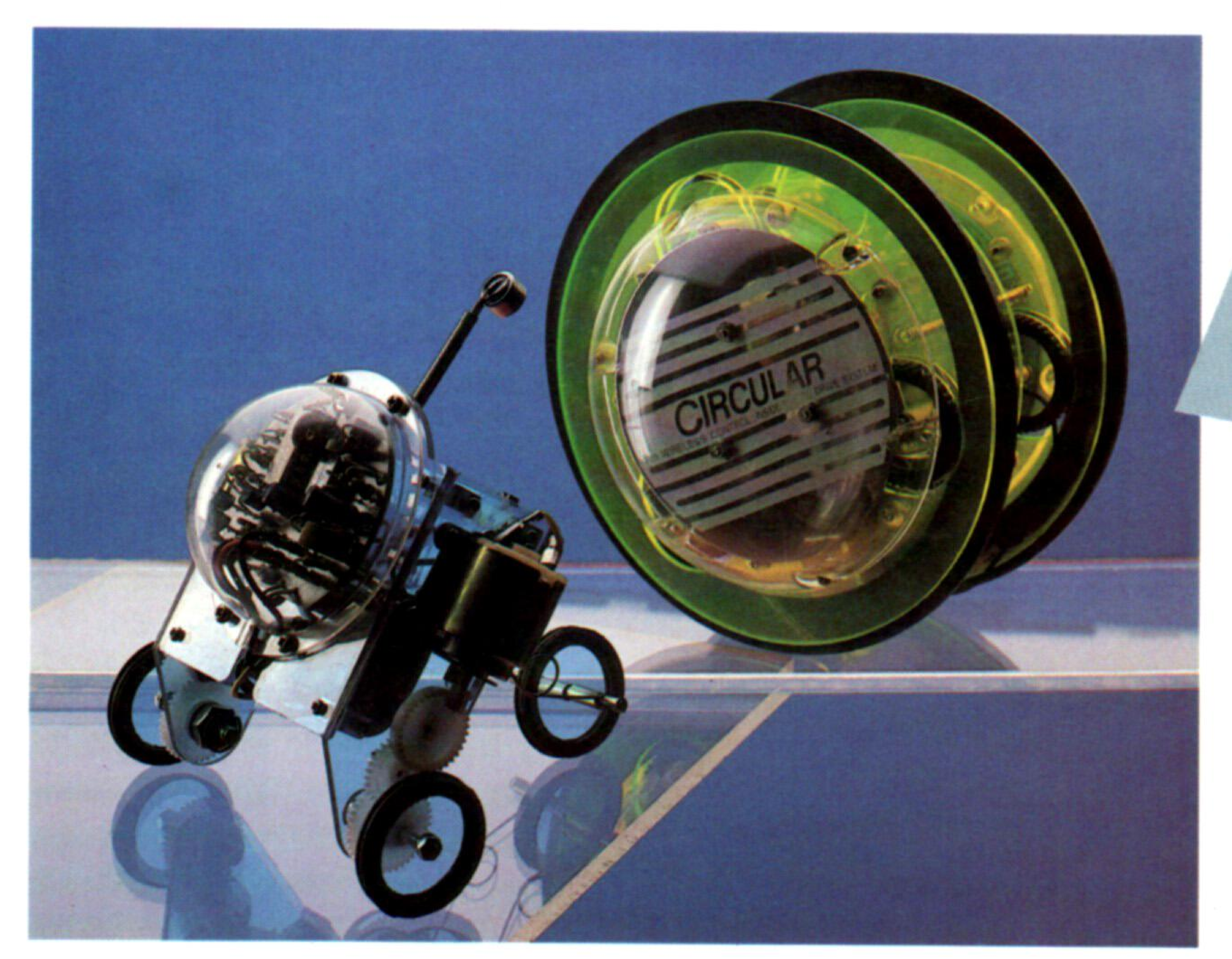
Muchos especialistas en educación piensan que uno de los beneficios que le reporta al niño la programación de robots es que ciertos "errores" se pueden eliminar de sus procesos mentales. La experiencia del ensayo y error, y el consiguiente desarrollo de un método de planificación más secuencial, hace que la condición del aprendizaje pase de una mentalidad de "bien o mal" al entorno más realista que requiere correcciones y modificaciones constantes para adecuarse a la situación dada. Los maestros que hoy trabajan con robots ven en ellos el mismo potencial que a comienzos de los ochenta vieron en los ordenadores los maestros que comenzaron a incorporarlos. La aparición del LOGO y el C-LOGO, la caída de precios del hardware y el creciente interés comercial para la robótica por parte de las más importantes firmas informáticas le auguran un brillante futuro al empleo de los robots como medio auxiliar para la educación.

Jugando con la tortuga

El papel de la tortuga en la clase no se limita a la geometría, la trigonometría y la programación. Se han construido laberintos como parte de un proyecto sobre mitología (Teseo y el Minotauro) y sobre arquitectura religiosa (el laberinto de la catedral de Reims). Un proyecto particularmente ambicioso supuso la construcción de una ciudad a escala en las clases de arte y artesanía, que la tortuga debía recorrer, visitando varias tiendas diferentes para "adquirir" productos incluidos en su "lista de la compra".
Otros juegos con la tortuga son *Racetrack*, en el que cada niño debe competir con los otros en programarla para que complete el recorrido de una pista de carreras sencilla, y *Postman*, en el que los niños se sientan en círculo y se envían entre sí mensajes escritos, que reparte la tortuga.
Turnturtle y *Shoveturtle* son dos juegos diseñados para ayudar a enseñar el desplazamiento angular y la distancia. En el primero, la tortuga se coloca en el centro de un círculo dividido en secciones. Cada sección se señala con un número diferente (marcador) y el niño ha de programar la tortuga para que gire y se detenga señalando hacia una de las secciones. Gana quien obtenga el marcador más alto con la menor cantidad de instrucciones. *Shoveturtle* utiliza un método similar pero siguiendo una pista recta.



Tanque programable
El Big Trak es una unidad autocontenida que se puede programar al estilo LOGO utilizando su teclado numérico. Las instrucciones entradas se componen de una dirección (izquierda, derecha, etc.) seguida por la cantidad de pasos (hasta 99). Big Trak puede recordar y efectuar de forma secuencial hasta ocho instrucciones diferentes





Trío futurista
Aquí tenemos los tres "robots" más caros de la gama Movits. Son, de izquierda a derecha, el Piper Mouse, el Circular y el Memocon Crawler. Cada máquina de la gama se halla dentro de una atractiva carcasa de plástico transparente, que deja a la vista los mecanismos internos (los motores y los componentes electrónicos que activan los dispositivos) y le confiere a la máquina un adecuado aire "futurista"

Nombre: ¿robot?

Los Movits son "kits" de robot, económicos y preprogramados, de gran demanda en el floreciente mercado de este tipo de productos

La predicción general apunta a que la robótica será el próximo paso en la evolución hacia el hogar informatizado. La etapa de desarrollo que ha alcanzado en la actualidad la robótica se está comparando a la del microordenador a comienzos de los años setenta, con las tecnologías que van surgiendo combinándose entre sí para formar un producto enteramente nuevo. Mientras que el microordenador se desarrolló en una relativa oscuridad, con apenas un puñado de aficionados a la electrónica comprendiendo el potencial de las nuevas máquinas, en la actualidad son muchas más las personas conscientes de la actual revolución de la microelectrónica.

Dado el amplio interés que existe por esta tecnología, era sólo cuestión de tiempo que algún fabricante emprendedor apareciera con una serie de "robots" de costo reducido para el hogar. Los Movits, fabricados por la empresa japonesa Kaho Musen, figuran entre los primeros intentos por satisfacer y rentabilizar el interés del público por la robótica.

Cada Movit viene en forma de modelo para armar (*kit*), sin que sea necesario ningún componente extra, aparte de las pilas de 1,5 V, e incluye un destornillador para facilitar su montaje. Rodeadas por una carcasa de plástico sólido con una cúpula de plástico transparente que deja a la vista sus sistemas de circuitos, las máquinas poseen un sugerente aspecto futurista. En la actualidad hay cinco Movits a la venta, siendo el más barato de todos el Monkey (mono), controlado por sonido. Al activarse a través del sensor de sonido ("escuchando" un grito o una palmada, p. ej.), dos brazos de movimiento alterno empujan el aparato a lo largo de una cuerda o algún otro objeto estrecho. El sensor es un transistor de micrófono condensador conectado mediante una placa de circuito impreso, que decodifica la señal. Las señales apropiadas harán arrancar el motor eléctrico CD, que hace girar un mecanismo de manivela y provoca el movimiento de los brazos. Un reloj situado en la placa asegura que el Monkey se detenga.

El segundo Movit en la escala de precios es el Line Tracer II. Se controla mediante un sensor infrarrojo fijado a su base, y sigue cualquier línea de gran contraste que se dibuje en el suelo. Este robot es impulsado mediante dos motores CD que generan la potencia para sus tres ruedas. A continuación viene el Pier Mouse, también controlado mediante un sensor de sonido, gobernado esta vez por un "silbato para perros". Siguiendo cada soplido del silbato, el Movit ejecuta una instrucción de una secuencia de instrucciones preestablecidas: girar izquierda, parar, girar derecha, parar, avanzar, parar. Siguiendo esta secuencia, el Movit se desplaza en línea recta.

El Circular Movit es la primera de las máquinas con movimiento gobernado completamente a control remoto. Activado por radio, lo impulsan dos motores CD que activan las grandes ruedas de la parte exterior de las burbujas. En la caja que contiene los mandos hay dos botones: cada uno de ellos gobierna uno de los motores. Por consiguiente, para hacer girar el Movit se han de pulsar bien el botón izquierdo, bien el derecho, y ambos botones a la vez para que se desplace hacia adelante. Para el control independiente hay dos placas PCB, cada una de las cuales toma una señal diferente de la caja de control.

Los dispositivos que hemos descrito hasta ahora son atractivos e interesantes de construir, pero la tecnología que utilizan apenas puede describirse como avanzada. El más "vanguardista" de estos robots es el Circular, pero incluso esta tecnología ya está siendo utilizada desde hace algún tiempo, como, por citar algunos ejemplos bien visibles, en los modelos a escala de barcos y aviones controlados por radio que se pueden ver en los parques de las ciudades de todo el mundo. La cualidad esencial

de lo que entendemos por robot es que lleve incorporado algún sistema de guía que se pueda programar para llevar a cabo una secuencia de acciones.

El Memocon Crawler es una máquina que se acerca mucho a nuestra definición de lo que es un robot y, al igual que otros Movits, está alimentado por dos motores eléctricos CD. El sistema de guía se compone de un teclado conectado al Crawler mediante un cable plano. El teclado posee cinco interruptores separados por cada una de las cinco instrucciones disponibles: adelante, izquierda, derecha, pausa y hacer sonar un zumbador e iluminar los LED de la máquina. Estas instrucciones se almacenan en un chip de RAM estática con una memoria que contiene 256 bytes, constando cada byte de cuatro bits. Los bytes que retienen las instrucciones no son direccionales de forma individual y se debe acceder a ellos en secuencia. Esto significa que no se pueden repetir pasos dentro de la secuencia, aunque, por supuesto, se puede volver a ejecutar toda la secuencia entera otra vez.

Dado que vienen en forma de modelo para armar y es necesario montarlos, a muchas personas no les interesará comprarse un Movit. Sin embargo, a pesar de la pasmosa cantidad de componentes que se proporcionan con cada *kit*, en realidad la construcción de las máquinas es notablemente sencilla, e incluso los niños pueden montarla. De hecho, un alto ejecutivo de una de las principales empresas distribuidoras hizo hincapié en que su hijo, de diez años de edad, era el principal montador de los modelos de demostración de la empresa.

Los diversos componentes de los robots vienen en bolsas separadas que llevan marcado un número. Las instrucciones consisten en un esquema del Movit. Cada pieza está dibujada por separado e identificada claramente por el número de su bolsa; mediante flechas se señala con exactitud dónde va cada componente. A la mayoría de las personas esta ayuda les resulta suficiente para completar el robot; pero hay, además, instrucciones escritas que le serán de ayuda en cualquier etapa de la construcción que le resulte confusa. Puesto que la placa de circuito impreso ya está montada, las únicas herramientas que se requieren son un par de llaves, un destornillador y un pequeño martillo.

Construir los Movits es, qué duda cabe, divertido, y los dueños obtendrán muchísima satisfacción al ver moviéndose por el suelo un robot que ellos mismos han ensamblado. Lo que ya es más dudoso es si los Movits representan realmente una entrada en el mundo de la robótica. No obstante, son económicos y, aunque no enseñen gran cosa sobre robótica, podrían proporcionarles a algunas personas, en especial a los niños, una iniciación en este tema, tan interesante y cada vez más popular.

Rodeado por círculos
El Circular, al igual que los otros integrantes de la gama, es fácil de construir. Las placas de circuito impreso para el robot propiamente dicho y la caja de control remoto ya están montadas. Esto significa que el proceso de construcción no exige soldaduras y que quien desee montar su propio Movit no necesitará conocimientos de electrónica para llevar a cabo su propósito

Chris Stevens



MONKEY

MOVIMIENTO
Dos brazos prensores que se mueven alternativamente, activados por movimiento de manivela

CONTROL
Sensor de sonido

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V



LINE TRACER II

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Sensor infrarrojo

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V

PIPER MOUSE

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Sensor de sonido

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V

CIRCULAR

MOVIMIENTO
Dos ruedas con engranaje de dientes activadas por dos motores CD

CONTROL
Caja de control por radio

FUENTE DE ALIMENTACION
Tres pilas de 1,5 V, una pila de 9 V y otra pila de 9 V para la caja de control

MEMOCON CRAWLER

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Memoria programable

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V



Lista de mercancías

El Nuevo Mundo
La era de los grandes descubrimientos se inicia a fines del s. XV. En el s. XVI, los exploradores europeos estaban apenas empezando a elaborar los mapas de los continentes. Al principio estas cartas de navegación eran esquemáticas y a menudo las islas (e incluso continentes enteros) se colocaban en lugares equivocados. A medida que vaya avanzando nuestra simulación, iremos revelando una parte diferente del mapa del mundo, tal como los primeros exploradores fueron uniendo entre sí las piezas de las masas de tierra del globo

Ahora necesitamos adquirir algunos artículos más, indispensables para el éxito de la expedición

Los salarios de la tripulación se abonarán sólo si el barco regresa a puerto, y en ese momento. El jugador debe decidir cuánto invertir en mercancías para comerciar, con la esperanza de obtener un beneficio, y cuánto oro dejar en reserva (o no) para ayudar a pagar los salarios de la tripulación cuando se regrese. Las mercancías disponibles son medicinas, armas, sal, tela, cuchillos y joyas; y se informa al jugador sobre sus precios y se le pregunta la cantidad a comprar. Al cabo de cada transacción se imprime el balance de dinero. Si no hay suficientes fondos para pagar el pedido, así se le informará al jugador y se le solicitará que vuelva a probar. La bodega del barco tiene capacidad ilimitada.

Al igual que en las dos etapas de preparación previas, las matrices para la tercera etapa, que representan los nombres, precios y cantidades de las mercancías compradas, se DIMensionan al comienzo del programa. Las mismas se añaden a las sentencias de inicialización anteriores, comenzando en la línea 30. La cantidad de cada tipo adquirido se retendrá en la matriz OA(). Las descripciones de las otras mercancías están retenidas en la matriz D$() DIMensionada en la línea 31. Los elementos de esta matriz se establecen en las dos líneas siguientes. El primero, D$(1), se establece como FRASCO DE MEDICINA. Los cinco elementos restantes, de D$(2) a D$(6), se establecen en ARMA, BOLSA DE SAL, BALA DE TELA, CUCHILLO y JOYA. Como antes, la referencia a un objeto por su dirección en la matriz facilitará la subsiguiente programación.

El costo de estas mercancías se DIMensiona en una matriz, OC (), que contiene seis elementos, en las líneas 34 y 35. El primer elemento, OC(1), representa el precio de las medicinas, que valen una pieza de oro el frasco. Las armas valen cinco piezas de oro (OC(2)=5), y con una pieza de oro se pueden comprar cinco bolsas de sal (OC(3)=0.2). Una bala de tela cuesta 2 piezas de oro (OC(4)=2), los cuchillos, media pieza de oro (OC(5)=0.5), y las joyas, una pieza de oro cada una (OC(6)=1).

En el programa principal, la línea 500 llama a la rutina para contratar la tripulación, y la línea 550 llama a la subrutina de aprovisionamiento. Ahora se debe insertar la línea 600 para llamar a la subrutina que se encarga de la adquisición de las otras mercancías. Veamos de forma detallada esta subrutina (líneas 3000-3999). Tras limpiar la pantalla hay una demora, y las líneas 3005 y 3010 imprimen y subrayan el título. Se le dice entonces al jugador que para el viaje se necesitarán otras mercancías.

La programación para la compra de los seis tipos de mercancías está organizada en un bucle. El bucle se utiliza una vez para cada tipo, empleando T como el contador de bucle de 1 a 6. Se le dice al jugador el costo de cada producto y su nombre se

Ian McKinnell

representa mediante D$(T) en la línea 3070. La primera vez que se realice el bucle, T se establecerá en 1, para el primer producto: medicinas. El costo por unidad del producto comercial actual se establece mediante OC(T) en la línea 3080. Cuando T se establece en 1, OC(T) representará una pieza de oro, que es el costo de un frasco de medicina. La segunda vez que se ejecute el bucle, T se habrá incrementado a 2, y D$(T) se establecerá en ARMA y el costo, OC(T), será cinco piezas de oro, y así sucesivamente hasta que T alcance el valor 6.

En función de si el costo es de una o más piezas de oro, se imprime el mensaje en singular o en plural, en las líneas 3085 y 3086. A continuación se produce una demora y se le pregunta al jugador: "QUIERES COMPRAR? (S/N)". La línea 3110 espera una respuesta, P$. La línea 3120 comprueba el carácter situado más a la izquierda de la respuesta. Si es "N" se lo envía a la línea 3175, enviando el programa hacia atrás hasta el comienzo del bucle, para presentarle el siguiente producto. Si la entrada es "S" o "SI", hay una demora y la línea 3130 pregunta "CUANTO QUIERES?". La línea 3135 espera una respuesta y el jugador puede responder digitando 10, o 10 BOLSAS DE SAL. Esta elección es posible porque la línea 3140 examina la respuesta y emplea VAL para reconocer los caracteres numéricos al comienzo de la frase e ignorar el texto.

Si usted trata de hacer una compra ilegal, sin tener oro suficiente para pagar la factura, la línea 45 hace una comprobación para determinar si el costo del producto, OC(T), multiplicado por el número entrado, TT, es mayor que el dinero disponible, MO. De ser así, se envía el programa a la línea 3150, que le informa al jugador acerca de su insolvencia monetaria. Tras una demora, se le dice al jugador "POR FAVOR, VUELVE A ENTRAR", en la línea 3154. El programa retorna a la línea 3130, que pregunta "CUANTO QUIERES COMPRAR?" si es que está en condiciones de abonar el nuevo pedido. Si hay fondos suficientes, el programa va a la línea 3160, donde la cantidad gastada se resta del fondo, mediante la fórmula: MO=MO – (OC(T)*TT). MO representa el dinero restante. El costo del producto, OC(T), se multiplica por la cantidad comprada, TT, y se resta del balance. TT se almacena en la matriz OA() en la línea 3165 y, tras una demora, la línea 3176 imprime una línea en blanco y el balance de oro. Tras otra demora, T se incrementa en 1 y el programa retorna al comienzo del bucle mediante la línea 3200, para ofrecer el siguiente producto.

Una vez seleccionados todos los productos (y al cabo de una pausa) la línea 3210 le dice al jugador que la etapa de compra ha terminado. La línea 3230 imprime "PULSA CUALQUIER TECLA PARA CONTINUAR" utilizando la variable establecida al comienzo del programa, K$, y aguarda una respuesta antes de retornar al programa principal. La tarea final consiste en resumir las mercancías pedidas y la tripulación contratada antes de pasar al inicio del viaje. Esto lo hace la rutina Resumen de Apertura.

El programa imprime "CUENTAS CON LA SIGUIENTE TRIPULACION" y se imprimen en un bucle las categorías de tripulante, en la línea 645 (FOR T=1 TO 5) para cada una de las cinco categorías de la tripulación. La línea 650 comprueba si se ha contratado alguno de esta categoría, comprobando CC(T), el contador de la tripulación para ese tipo. Si el contador es 0, la línea 670 envía el programa hacia atrás, hasta el comienzo del bucle, incrementando en uno el valor de T. Si se ha contratado a algún tripulante de esa categoría, se imprime la cantidad y la categoría. Tras la descripción se imprimen una S o un espacio, realizando el programa cinco veces el bucle, una para cada categoría de tripulante. De modo similar se imprime una lista de las provisiones, entre las líneas 675 y 710, utilizando un bucle de 4 (línea 685) para las cuatro clases de provisiones.

El resto de las mercancías adquiridas no se imprime en un bucle porque sus descripciones son muy diferentes. Éstas no se describen por unidades, por lo cual la simple adición de una "S" no creará necesariamente una palabra en plural; una bala de tela se convierte en bala*s* de tela, por ejemplo. Cada una de las mercancías extras se trata por separado de forma lineal. Si no se ha comprado nada de algún tipo de mercancía, el programa pasa directamente a la siguiente categoría. La línea 730, por ejemplo, comprueba si se han comprado medicinas y, si no, el programa salta hasta la sección siguiente, en la línea 740. Si se han comprado medicinas,



Las mercancías esenciales
Éstas son las mercancías que nuestros expedicionarios podrían haber esperado llevar consigo para comerciar con los habitantes de las nuevas tierras. Por supuesto, también habrán tenido que abonar las mercancías antes de hacerse a la mar con ellas. En nuestra simulación, los nombres de cada una de las posibles mercancías transportadas están retenidos en la matriz D$. El precio por unidad está retenido en la matriz OC, mientras que la cantidad adquirida está retenida en la matriz OA

Ian McKinnell

el programa determina si fueron un frasco o más, e imprime la terminación adecuada: una S o un espacio. Luego imprime la cantidad y pasa a la siguiente categoría. Una vez completada la lista, la línea 792 informa al jugador sobre la cantidad de dinero que le queda. Se le dice al jugador "PULSA CUALQUIER TECLA PARA COMENZAR EL VIAJE" en la línea 797, y entonces el programa espera alguna entrada.

Módulo Tres: Mercancías para comerciar

Dimensionamiento de las matrices

```
30 DIM OA(6)
31 DIM D$(6)
32 D$(1)="FRASCO DE MEDICINA":D$(2)="ARMA":D$(3)=
   "BOLSA DE SAL"
33 D$(4)="BALA DE TELA":D$(5)="CUCHILLO":D$(6)=
   "JOYA"
34 DIM OC(6)
35 OC(1)=1:OC(2)=5:OC(3)=.2
36 OC(4)=2:OC(5)=.5:OC(6)=1
```

Llamada a Subrutina Mercancías para Comerciar

```
600 GOSUB 3000
```

Rutina Resumen de Apertura

```
605 REM **** LISTO PARA EMPEZAR ****
610 PRINTCHR$(147)
615 S$="AHORA ESTAS LISTO PARA EMPEZAR*":GOSUB 9100
625 S$="EL VIAJE.*":GOSUB 9100
630 GOSUB 9200
635 PRINT:S$="CUENTAS CON LA SIGUIENTE
    TRIPULACION:*":GOSUB 9100
640 GOSUB 9200
645 FOR T=1 TO 5
650 IF CC(T)=0THEN 670
655 PRINT CC(T);
660 PRINT C$(T);
662 IF CC(T)=1THEN PRINT" ":GOTO 668
664 PRINT "S"
668 GOSUB 9200
670 NEXT
674 GOSUB 9200
675 PRINT:S$="Y LAS SIGUIENTES PROVISIONES:*":GOSUB
    9100
680 GOSUB 9200
685 FOR T=1 TO 4
690 IF PA(T)=0 THEN 710
695 PRINT PA(T);U$(T);"S DE";
700 PRINT P$(T)
708 GOSUB 9200
710 NEXT
715 GOSUB 9200
720 PRINT:S$="TAMBIEN POSEES:*":GOSUB 9100
725 GOSUB 9200
730 IF OA(1)=0 THEN 740
733 IF OA(1)=1 THEN S$="FRASCO DE MEDICINA*":GOTO 735
734 S$="FRASCOS DE MEDICINA*"
735 PRINT OA(1);:GOSUB 9100
736 GOSUB 9200
740 IF OA(2)=0THEN 750
743 IF OA(2)=1 THEN S$="ARMA*":GOTO 745
744 S$="ARMAS*"
745 PRINT OA(2);:GOSUB 9100
746 GOSUB 9200
750 IF OA(3)=0 THEN 760
753 IF OA(3)=1 THEN S$="BOLSA DE SAL*":GOTO 755
754 S$="BOLSAS DE SAL*"
755 PRINT OA(3);:GOSUB 9100
756 GOSUB 9200
760 IF OA(4)=0 THEN 770
763 IF OA(4)=1 THEN S$="BALA DE TELA*":GOTO 765
764 S$="BALAS DE TELA*"
765 PRINT OA(4);:GOSUB 9100
766 GOSUB 9200
770 IF OA(5)=0 THEN 780
773 IF OA(5)=1 THEN S$="CUCHILLO*":GOTO 775
774 S$="CUCHILLOS*"
775 PRINT OA(5);:GOSUB 9100
776 GOSUB 9200
780 IF OA(6)=0 THEN 790
783 IF OA(6)=1 THEN S$="JOYA*":GOTO 785
784 S$="JOYAS*"
785 PRINT OA(6);:GOSUB 9100
786 GOSUB 9200
790 GOSUB 9200
792 PRINT:PRINT"TE QUEDAN ";MO;"PIEZAS DE ORO"
796 GOSUB 9200
797 S$="PULSA CUALQUIER TECLA PARA COMENZAR EL
    VIAJE*"
798 GOSUB 9100
799 GET P$:IF P$=""THEN 799
999 END
```

Subrutina Mercancías para Comerciar

```
3000 REM **** ETAPA 3 OTRAS MERCANCIAS ****
3001 PRINTCHR$(147):REM ETAPA 3
3002 GOSUB 9200
3005 PRINT"     ETAPA 3 - OTRAS MERCANCIAS"
3010 PRINT"     ----- - - ----- ----------"
3020 GOSUB 9200
3025 PRINT
3030 S$="HAY OTRAS COSAS QUE PODRIAN*":
     GOSUB 9100
3035 S$="SERIE UTILES PARA EL VIAJE,POR*":
     GOSUB 9100
3040 S$="EJEMPLO MEDICINAS Y MERCANCIAS*":
     GOSUB 9100
3045 S$="PARA COMERCIAR.*":GOSUB 9100
3046 GOSUB 9200
3050 S$="PUEDES NECESITAR TAMBIEN ESCOPETAS.*":
     GOSUB 9100
3055 GOSUB 9200:GOSUB 9200
3060 FOR T=1 TO 6
3065 PRINT
3070 PRINT"UN";D$(T);
3075 S$="CUESTA*":GOSUB 9100
3080 PRINT OC(T);
3085 IF OC(T)=1 THEN PRINT "PIEZA DE ORO":GOTO
     3090
3086 PRINT "PIEZAS DE ORO"
3090 GOSUB 9200
3095 S$="TE GUSTARIA COMPRAR (S/N)*":GOSUB 9100
3110 INPUT P$:P$=LEFT$(P$,1)
3115 IF P$<>"S" AND P$<>"N" THEN 3095
3120 IF P$="N" THEN 3175
3125 GOSUB 9200
3130 S$="CUANTO QUIERES*":GOSUB 9100
3135 INPUT P$
3140 TT=VAL(P$)
3145 IF OC(T)*TT>MO THEN 3150
3147 GOTO 3160
3150 S$="NO TIENES DINERO SUFICIENTE*":GOSUB 9100
3152 GOSUB 9200
3154 S$="POR FAVOR VUELVE A ENTRAR*":GOSUB 9100
3155 GOSUB 9200:GOTO 3130
3160 MO=MO - (OC(T)*TT)
3165 OA(T)=TT
3170 GOSUB 9200
3175 PRINT
3176 PRINT" DINERO SOBRANTE = ";MO
3200 GOSUB 9200:NEXT T
3205 GOSUB 9200:PRINT:PRINT
3210 S$="FIN DE LA ETAPA 3*":GOSUB 9100
3220 GOSUB 9200:PRINT
3230 S$=K$:GOSUB 9100
3240 GET P$:IF P$=""THEN 3240
```

Complementos al BASIC

Spectrum:
Realice las siguientes modificaciones:

```
32 DIM D$(6,20)
610 CLS
799 LET P$=INKEY$:IF P$="" THEN GO TO 799
3001 CLS
3110 INPUT P$:LET P$=P$(1 TO 1)
3240 LET P$=INKEY$: IF P$="" THEN GO TO 3240
```

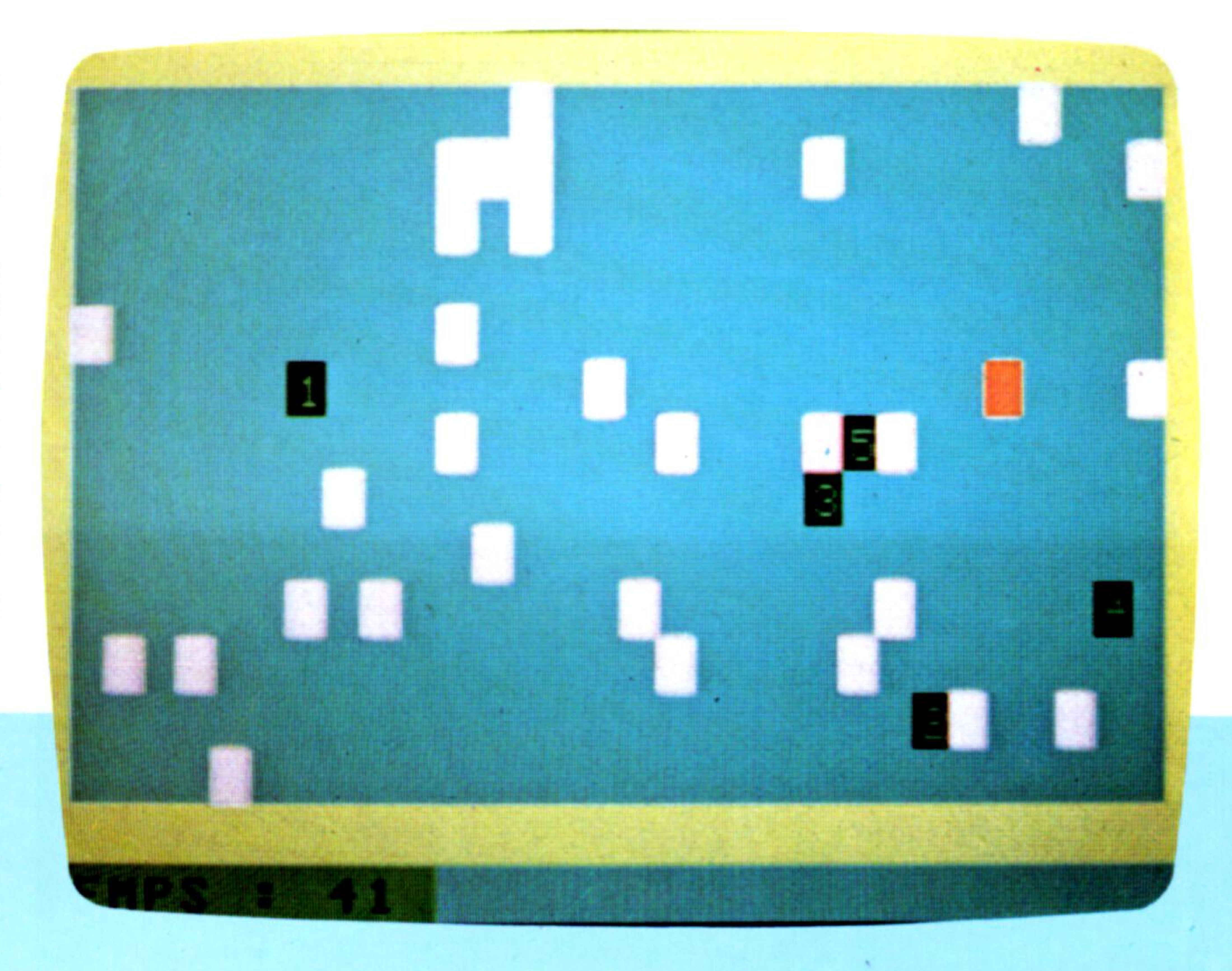
BBC Micro:
Realice las siguientes modificaciones:

```
610 CLS
799 A$=GET$
3001 CLS
3240 A$=GET$
```

Cifras

He aquí un juego de destreza más difícil de lo que parece. Esta versión es para el microordenador Alice, de Matras

Usted debe tratar de obtener la mayor cantidad de puntos posible al borrar, con la ayuda del cuadrado rojo, las cifras que muestra el ordenador. Las teclas a utilizar son W (abajo), Q (izquierda) y S (derecha). La barra espaciadora le permite detenerse. *Atención:* Al borrar los números debe seguir un orden (de menor a mayor), evitando los obstáculos colocados al azar en la pantalla, y hacerlo en un período de tiempo limitado. (El tiempo que le va quedando se visualiza en la última línea.) Cuando todas las cifras hayan sido borradas, el juego se reanuda con un número más. Al pasar de las nueve cifras se añade una dificultad: debe borrar los símbolos acordándose del orden en que han aparecido (¡a no ser que no se sepa de memoria el código ASCII!).



```
  5 REM *************
 10 REM *   CIFRAS   *
 15 REM *************
 20 S=0
 30 X=0
 40 P$=CHR$(191)
 50 GOSUB 2000
100 $FOR I=1 TO X
110 D$=INKEY$
120 D(D$="Q")-(D$="S")+32*((D$=
    "Z")-(D$="W"))
130 IF D<>0 THEN D0=D
140 IF D$=" " THEN D0=0
150 T=T-0.1
160 PRINT@ 480, "TIEMPO :";INT(T+1);
170 IF T<0 THEN 500
180 P=P+D0
190 C=PEEK(16384+P)
200 IF C=I+48 THEN S=S+I:SOUND 1,1: GOTO 260
210 IF C<>223 THEN P=P1
220 PRINT@ P1,CHR$(223);
230 PRINT@ P,P$;
240 P1=P
250 GOTO 110
260 PRINT@ P1,CHR$(223);
270 PRINT@ P,P$;
280 P1=P
290 NEXT I
300 GOSUB 2000
310 GOTO 100
500 IF R<S THEN R=S
510 PRINT@ 166,"TIEMPO TRANSCURRIDO";
520 PRINT@ 234, "PUNTOS :";S;
530 PRINT@ 266, "RECORD :";R;
540 PRINT@ 326, "OTRA ?";
550 D$=INKEY$
560 IF D$="" THEN 550
570 IF D$<>"N" THEN 20
580 END
2000 CLS 6
2010 X=X+1
2020 FOR I=0 TO 31
2030 PRINT@ I,CHR$(159);
2040 PRINT@ 448+I,CHR$(159);
2050 NEXT I
2060 FOR I=1 TO 13
2070 PRINT@ I*32,CHR$(159);
2080 PRINT@ I*32+31,CHR$(159);
2090 NEXT I
2100 FOR I=1 TO 30
2110 GOSUB 3000
2130 PRINT@ P,CHR$(239);
2140 NEXT I
2150 FOR I=1 TO X
2160 GOSUB 3000
2180 POKE 16384+P, I+48
2190 NEXT I
2200 GOSUB 3000
2210 PRINT@ P,P$;
2220 P1=P
2230 T=50
2240 D0=0
2250 RETURN
3000 P=RND(414)+32
3010 IF PEEK(16384+P)<>223 THEN 3000
3020 RETURN
```

Amplios horizontes

El sintetizador Music 500 incorpora una tecnología que hasta ahora no se hallaba al alcance de los micros personales



Chris Stevens

Música de calidad
El paquete Music 500 de Hybrid es, ciertamente, uno de los más avanzados dispositivos de música por ordenador que se han creado para micros personales. Equipado con 16 osciladores y modulación de onda digital, puede competir ventajosamente con sintetizadores mucho más caros

El Music 500 Package, comercializado por Acorn para el BBC Micro, fue diseñado por otra empresa de Cambridge, Hybrid Technology. Se trata de una compleja pieza electrónica alojada en una caja de unidad de disco de color crema. Se conecta al conector para bus de 1 MHz del BBC Micro y a un amplificador estéreo a través de un conector DIN estándar de 5 patillas.

En el interior de la caja hay 16 osciladores, pudiendo el ordenador programar la frecuencia (*pitch*), volumen, equilibrio y forma de onda de cada uno de ellos. Con el software que lo acompaña, el Music 500 supera en rendimiento a numerosos sintetizadores que existen en la actualidad en el mercado y cuyo precio es varias veces superior.

La mayoría de los sintetizadores producen su escala de sonidos por uno de dos métodos. Los más económicos toman un tono simple (una onda senoidal) y luego la distorsionan y la filtran para producir el efecto requerido. Aunque este método puede producir una gama de sonidos razonable, es aún muy limitado, incluso en las manos de un experto. Algunos de los sintetizadores más caros utilizan varias ondas senoidales que se modulan las unas a las otras. La amplitud de una onda se emplea para controlar la frecuencia de otra. Este método de control es muy superior al método de distorsión simple, pero aun así no puede imitar con exactitud la mayoría de los sonidos producidos naturalmente.

El Music 500, sin embargo, produce sus sonidos por un procedimiento completamente distinto. Las formas de onda creadas por este dispositivo se construyen de forma digital para conformar un patrón y se muestrean a una velocidad de 47 000 veces por segundo. Además de esta capacidad para programar cualquier forma de onda, el Music 500 ofrece técnicas de modulación; no obstante, no se limita a la modulación de frecuencia, y el Music 500 puede combinar canales con modulación de anillo y modulación de fase. También puede utilizar otras formas de onda además de las ondas senoidales simples.

Si no fuera por el software que se proporciona con el Music 500, la simple creación de un tono con este dispositivo sería una tarea difícil. Este software se suministra en cassette, si bien se lo puede transferir fácilmente a disco con el programa que se proporciona. Hybrid Technology tiene planeado lanzar próximamente una versión ampliada del software en ROM. El paquete se suministra con AMPLE, un lenguaje de programación de música que ofrece todo lo necesario para sacar el máximo partido del Music 500. La utilización del lenguaje es, en muchos sentidos, similar al FORTH y al LOGO, tratándose de un lenguaje basado en palabras con las cuales los programadores pueden definir palabras clave y procedimientos. Una composición musical podría tener más o menos el siguiente aspecto:

```
10 .%EL VUELO DEL ABEJORRO
20 .preparacion
30 .ejecucion
```

El vocablo *preparacion* es una palabra definida por el usuario que ya se ha programado del modo siguiente:

```
10 .'preparacion'
20 .[
30 .estsint
40 .estbajo
50 .esttambores
60 .estcimbalo
70 .]
```

Los términos con los que se define preparacion ya han sido programados, a su vez, para definir sonidos diferentes; por ejemplo *esttambores* o *estcimbalo*. Se proporcionan formas de onda preestablecidas para ayudarlo a iniciarse con el Music 500; sin embargo, es muy fácil crear formas de onda nuevas. Las formas de onda se pueden definir ya sea en términos de las proporciones relativas de los primeros 16 armónicos del tono, o bien generar en forma matemática. Este último método le otorga al Music 500 la capacidad de producir "ruido" especificando una forma de onda al azar. La definición para el sonido del címbalo, por ejemplo, podría ser la siguiente:

```
10 .'cimbalo'
20 .[
30 .3 WMOD 0 rand! 128 FOR(RAND? INDEX WG!)
   FOR WGC
40 .1 CHAN
50 .3 WAVE UN RM - 1 POS
60 .2 CHAN
70 .3 WAVE 2000 OFFSET 1 POS
80 .]
```

El equivalente de AMPLE para el bucle FOR...NEXT del BASIC establece cada punto de la forma de onda en un valor al azar. Esta forma de onda se utiliza para dos canales productores de sonido (dos osciladores) y el tono del segundo está ligeramente desplazado respecto al primero y en posición estéreo diferente. El resultado es un sonido ruidoso, estrepitoso... como el de un címbalo.

Como alternativa, se podría definir una forma de onda más melodiosa:

```
10 .'sint'
20 .[
30 .WZERO 181 1 WH! 135 2 WH! 181 3 WH!
         62 4 WH! 76 5 WH! 37 6 WH!
         14 7 WH! 3 8 WH!
40 .WHG WGC 3
50 .]
```

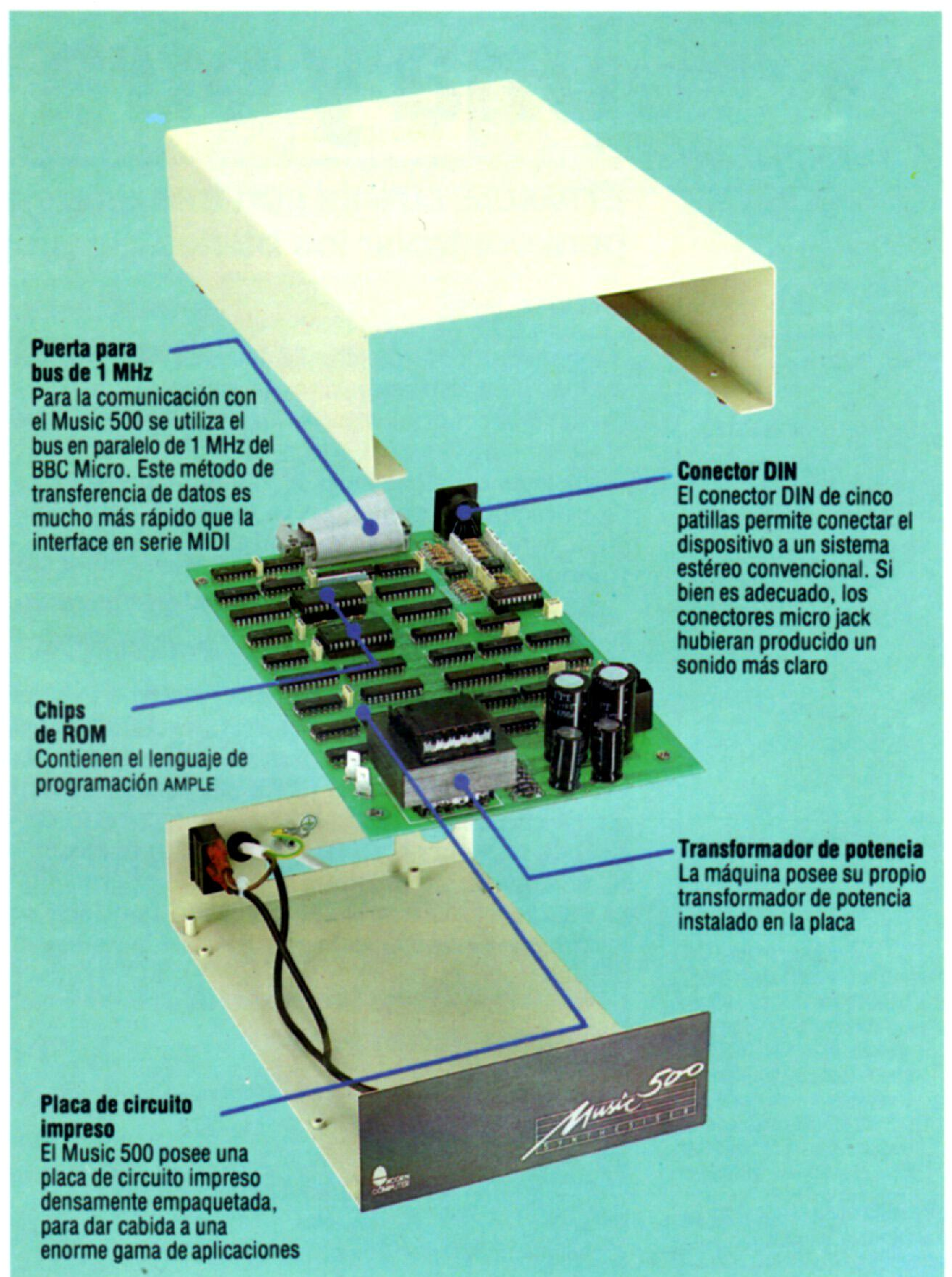
Esta forma de onda de compone de proporciones variables de los primeros ocho armónicos. Las envolturas de tono y amplitud se crean de forma similar, utilizando ya sea las formas simples por defecto o bien las que haya especificado el usuario.

La palabra ejecucion puede llamar a cada uno de los sonidos definidos y demás palabras que contenga la partitura actual.

```
10 ."ejecucion"
20 .[
30 .4 PLAYERS
40 .1 PLAY (melodia para piano)PLAY
50 .2 PLAY (linea de bajo)PLAY
60 .3 PLAY (ritmo de bateria)PLAY
70 .4 PLAY (estruendo de cimbalo)PLAY
80 .GO
90 .]
```

Para explicarlo de forma sencilla, se le da al sistema la cantidad de ejecutantes necesarios, a cada ejecutante se le da un sonido y una partitura (más palabras definidas por el usuario) y por último se les dice que empiecen a tocar, llevando el tiempo entre sí. Esta capacidad del AMPLE para tareas múltiples permite utilizar el Music 500 como cuando se escribe una partitura para orquesta. Uno no necesita instrumentar todas las partes al mismo tiempo, aunque podría hacerlo si lo deseara, y cada parte se trata por separado. Todo el lenguaje es modular y jerárquico. Incluso las palabras incorporadas se pueden volver a definir, aunque el empleo de esta facilidad es bastante complicado. No obstante, se puede, pongamos por caso, redefinir la palabra que marca las barras de modo que la segunda nota de cada barra se toque un poco más fuerte para dar acento.

Si todos estos sonidos son excesivamente técnicos, entonces el AMPLE proporciona algunas palabras de ayuda. En todo momento se ofrecen valores sensatos por defecto, de modo que si sólo desea tocar una melodía, puede limitarse a entrar las



Chris Stevens

notas por el teclado en lugar de dentro de una definición de palabra.

Acorn está por sacar un teclado musical completo de 49 notas que se conecta a la puerta para el usuario del BBC Micro, para utilizar con el Music 500. Éste incluirá software en AMPLE, como un "procesador de notas" que grabará lo que usted toque en el teclado y luego le permitirá volver a ejecutarlo y editarlo.

MUSIC 500

Osciladores	16
Gama de frecuencia	0-20 Khz
Resolución de tonos	16avo de semitono
Formas de onda	
Armónicas	16 armónicos
Geométricas	128 puntos
Envolturas (tono/amplitud)	17 segmentos
Longitud de paso	0-320 segundos
Resolución de tiempo	10 ms
Posiciones estéreo	7
Precisión de forma de onda	logarítmica 8 bits
Velocidad de muestreo	46875/seg
Resolución de frecuencia	0,0056 Hz
Base de tiempo	255 ms-655 ms/ciclo
Concurrencia	1-9 procesos

Una y otra vez

El PASCAL cuenta con tres diferentes construcciones de "iteración" para controlar los bucles del programa

Ya hemos analizado dos de los tres métodos del PASCAL para estructuración de sentencias: las estructuras secuenciales que utilizan las "palabras paréntesis" BEGIN y END, y la opción o selección, que emplea las construcciones IF y CASE. El tercer tipo de estructura de sentencias es la *iteración* o repetición, para la que el PASCAL proporciona tres construcciones diferentes. La sentencia FOR da un bucle "activado por contador", mientras que tanto las sentencias WHILE como REPEAT son "activadas por condición".

En primer lugar se evalúa la expresión booleana delimitada por las palabras reservadas WHILE y DO y, si el resultado de la expresión es verdadero, la sentencia siguiente (que puede ser cualquier sentencia en PASCAL, inclusive una sentencia estructurada de cualquier nivel de complejidad) se ejecutará repetidamente mientras la condición permanezca siendo verdadera. Esta condición booleana se vuelve a evaluar tras cada ejecución de la sentencia del "cuerpo" de la construcción WHILE. Obviamente, esto implica que las acciones incluidas en el bucle deben cambiar de alguna forma al menos uno de los elementos evaluados. Veamos un ejemplo:

```
WHILE MaxInt >1 DO
  WriteLn ('Buclando!')
```

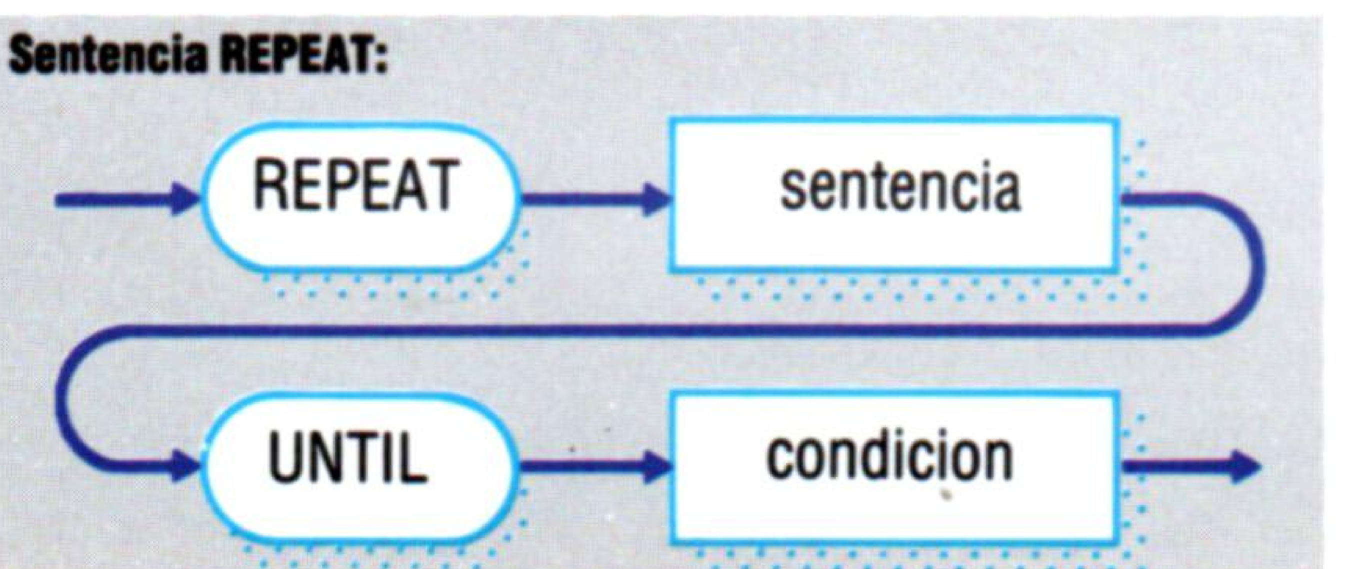
tendría enormes dificultades para terminar. Veamos un ejemplo más realista (y más seguro):

```
read (retirar);
WHILE retirar > SaldoBancario DO
  BEGIN
    WriteLn ('Fondos insuficientes — vuelve a
    probar:');
    write ('Cantidad?');
    read (retirar)
  END
```

Este segmento de programa ilustra muy bien una de las propiedades más valiosas de la construcción WHILE. Si, al leerse por primera vez la cantidad a retirar, la misma se halla dentro del saldo actual de la cuenta, el bucle WHILE no se ejecutará en absoluto. Si se ejecuta el cuerpo, la repetición se detendrá cuando la expresión resulte falsa. En este caso, el control del programa seguirá en la primera sentencia tras la construcción WHILE. El aspecto de la expresión booleana del encabezamiento de esta construcción indica sintácticamente la seguridad de la "semántica" (o el significado y comportamiento exactos de las sentencias) del PASCAL. Si usted desea iterar sólo cuando una condición es verdadera, habrá de utilizar el bucle WHILE.

A menudo hay ocasiones en las que necesitamos llevar a cabo el cuerpo de un bucle al menos una vez para preparar la condición que lo detenga, tal como ocurre con el programa Dorada que ofrecimos anteriormente como ejemplo. En estas circunstancias, la construcción REPEAT...UNTIL expresa con más precisión el algoritmo y proporciona una sintaxis natural de alto nivel.

Observe que ahora la condición de terminación aparece al final de la estructura, y que es lógicamente opuesta a la condición utilizada para codificar la construcción WHILE. Es decir, el bucle REPEAT termina cuando la evaluación de la expresión booleana es verdadera, mientras que un bucle WHILE (si es que hubiera comenzado) se detendrá cuando la condición sea falsa. Si el cuerpo de la sentencia REPEAT contiene varias sentencias, éstas, en realidad, deberían encerrarse entre las palabras paréntesis BEGIN...END, pero el PASCAL es elástico en cuanto a esta regla: las palabras reservadas REPEAT y UNTIL serán suficientes para delimitar el cuerpo del bucle. Es perfectamente legal incluir, sin embargo, las palabras superfluas BEGIN y END, algo así como usar un punto y coma extra antes de una palabra reservada, con la excepción de que no se cree ninguna sentencia nula.



Sentencia WHILE:
WHILE → condicion → DO → sentencia
condicion:= expresión booleana

Liz Dixon

El siguiente segmento contará la cantidad de veces que aparezca la letra "e" en una frase entrada desde el teclado. La frase debe terminar con un punto final, momento en el que termina el bucle REPEAT y se imprime el resultado.

```
contador:=0;
REPEAT
  read (simbolo);
  IF simbolo = 'e' THEN
    contador:=contador+1
UNTIL simbolo='.';
WriteLn ('En la frase hubo', contador: 1, '"e"s.')
```

Aquí no hay preferencia por ninguna construcción y podríamos haber usado un bucle WHILE:

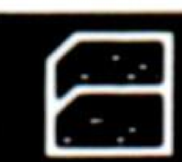
Respuesta al ejercicio Dorada
En el capítulo anterior planteamos un ejercicio con nuestro programa Dorada, que generaba términos Fibonacci y las proporciones de los pares sucesivos. Le sugerimos que tratara de modificar el bucle de modo que la condición de terminación se detuviera cuando el siguiente término Fibonacci a calcular fuera mayor que MaxInt. Con la representación del complemento a dos, si un resultado entero es mayor que MaxInt, el bit del signo pierde su significado. El número parece entonces negativo, de modo que el programa se podría romper, dado que el tipo Fibonacci excluye los valores negativos. Aun con enteros, el bucle no se detendría jamás. No podemos decir:

UNTIL primero + segundo > MaxInt

porque, por definición, no hay ningún número mayor que MaxInt. No obstante, una reacomodación algo más astuta para efectuar una resta en vez de una suma nos da la solución:

UNTIL segundo > MaxInt — primero

Ahora, por supuesto, ya no se requieren la aritmética real y la constante Epsilon



```
contador:=0;
read (simbolo);
WHILE simbolo <> '.' DO
  BEGIN
    IF simbolo = 'e' THEN
      contador:=succ(contador);
    read (simbolo)
  END;
write ('Hubo', contador : 1,);
WriteLn ('"e"s en la frase.')
```

Se puede ver, sin embargo, la diferencia existente entre las dos estructuras. La acción requerida para determinar la condición de terminación (la sentencia read, en este ejemplo) a menudo se producirá dos veces cuando se utilice una construcción WHILE, una vez inmediatamente antes de entrar a la construcción, y otra vez como la última sentencia del cuerpo del bucle WHILE.

Aunque el bucle WHILE es la única construcción esencial de todos los lenguajes, algunas veces es conveniente tener algún medio de repetir algo una cantidad específica de veces, normalmente a través de cierta gama de valores dentro de una escala. Para crear esta clase de bucles "activados por contador", el PASCAL ofrece una tercera construcción. Aquí el programador de BASIC podría sentirse en un terreno familiar, pero entre la construcción del PASCAL y el bucle FOR del BASIC existen algunas diferencias importantes que hay que tener en cuenta. En primer lugar, como controlador del bucle se puede utilizar cualquier variable escalar y no sólo enteros. En segundo lugar, y lo que es más importante, la sentencia FOR..DO del PASCAL es completamente segura; al igual que el bucle WHILE, puede directamente no ejecutarse (p. ej., si el valor inicial es mayor que el valor final) y existen severas restricciones en el uso de la variable controladora. En especial, es ilegal cambiar este valor en cualquier lugar del cuerpo del bucle. Además, ¡en PASCAL es un error el mero hecho de amenazar con cambiarlo! Cuando veamos procedimientos y funciones podremos apreciar la verdadera ventaja de estas medidas de seguridad; pero, mientras tanto, he aquí un ejemplo ilegal:

```
FOR N:=1 TO 10 DO
  IF N=10 THEN
    N:=1
```

Una soberana tontería, por supuesto. Habiendo solicitado que el bucle se realice diez veces, la sentencia IF volvería a restablecer el valor de N en 1 y crearía un bucle infinito.

Observe que la sintaxis de la construcción FOR..DO tiene una sentencia de asignación entre las dos primeras delimitadoras (FOR y TO) que le asignan el valor inicial a la variable de control del bucle, y el valor final está dado por la expresión encerrada entre las palabras reservadas TO y DO. Estos dos valores deben ser, por supuesto, del mismo tipo escalar que el controlador. Veamos otros ejemplos:

- FOR letra := 'A' TO 'Z' DO {etc}
- FOR mes := Ene TO Dic DO {etc}
- FOR N := N TO succ (MaxInt DIV 1000) DO {etc}

El último ejemplo da por sentado que en algún momento anterior a N ya se le ha asignado un valor; recuerde que si éste era mayor que el valor final especificado, el bucle no se llevaría a cabo. En aquellas ocasiones en las que deseamos descender en una escala de valores, en lugar de TO se utiliza la palabra reservada DOWNTO. De modo, que, por ejemplo, la NASA podría emplear:

```
FOR CuentaAtras :=10 DOWNTO 0 DO
  WriteLn (Cuentaatras : 32 — 3 *CuentaAtras);

WriteLn ('DESPEGUE!')
```

El PASCAL mantiene un control riguroso sobre el bucle FOR y no permite ningún equivalente del incremento y decremento STEP opcional del BASIC.

Sentencia FOR:

Programa Real Ale

```
PROGRAM        RealALe   ( output );

TYPE
  cerveza = (Mild, Bitter, Special, Warmer);
VAR
    precio,
    pintas        : real;
    ale           : cerveza;
BEGIN
  WriteLn ('Pintas' : 6,
        'Mild' : 9, 'Bitter' : 8,
          'Special' : 8, 'Warmer' :8);
  WriteLn;
  pintas :=0.5; {empezar por media pinta}

  REPEAT
    IF pintas - trunc (pintas) >0.4
      THEN {escribir como ##.#}
        write ( pintas : 6 : 1, ' ' )
      ELSE {escribir como un entero}
        write ( round ( pintas ) :4, '  ':3);

    FOR ale:= Mild TO Warmer DO
      BEGIN
        CASE ale OF
        Mild              : precio : =65;
        Bitter            : precio : =69;
        Special           : precio : =74;
        Warmer            : precio : =79
        END;    {CASE}
          {redondear y convertir a libras:}
        write ( round ( precio*pintas )
                          / 100 : 8 : 2 )
      END;
        {ahora empezar una nueva linea}
    WriteLn;
    pintas := pintas + 0.5

  UNTIL pintas >10
END.
```

¿A ti cuál te gusta?
El programa Real Ale imprime una lista de precios para cuatro clases distintas de cerveza para una gama de cantidades en incrementos de media pinta. Cada clase de cerveza se representa mediante un identificador de constante conceptual enumerado en la definición TYPE de cerveza. La selección del precio correspondiente se realiza naturalmente con una sentencia CASE. Las pintas enteras se imprimen como enteros rellenados con dos espacios extra, mientras que el formateo de las cantidades reales suprime las posiciones decimales no deseadas

Un brazo fuerte

Iniciamos nuestro proyecto proporcionando los patrones de las partes del brazo y la lista de componentes

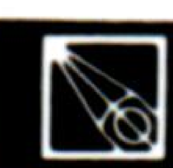
Los componentes necesarios para este proyecto probablemente habrán de obtenerse de diversas fuentes. Las tiendas de electrónica y de modelos a escala proporcionarán los componentes electrónicos y móviles, pudiéndose adquirir el resto en ferreterías.

La base y los miembros del brazo se deben construir con un material ligero y rígido que se pueda cortar y moldear fácilmente. Probablemente el material más accesible sea la madera contrachapada, aunque también se podría utilizar una lámina delgada de pérspex o un material plástico rígido. Suponiendo que para la estructura se emplee madera contrachapada, necesitaremos un cuadrado de 50 × 50 cm, de 4 mm de espesor. Lo usaremos junto con unos patrones para cortar los componentes de la estructura. Para actuar como bases de fijación necesitaremos tres trozos de madera de 55 mm (dos de 6 mm de espesor y uno de 12,5 mm de espesor).

Para el brazo necesitaremos cuatro servos digitales de 5 V; con este fin son ideales los motores Futaba S128. En el mismo sitio donde se adquieran los motores, se deben comprar discos plásticos de 30 mm para colocar en el extremo de los husillos

cortar 2
brazo inferior
cortar 2
brazo superior
cuerpo principal
cortar 2
pinza
cortar 2
pieza en T
de la pinza
cortar 1
base superior del
cuerpo principal
cortar 1

Kevin Jones



Cortado de los patrones
Haga calcos de los patrones para los componentes de la estructura, fíjelos sobre el tablero de contrachapado y córtelos. Observe que algunos patrones se usan dos veces para producir pares de componentes. Empareje los pares de los brazos y únalos antes de taladrarlos. Lije con cuidado cada uno de los componentes para eliminar las asperezas de los bordes

del motor. El mejor sitio tal vez sean las tiendas de modelos a escala. Si desea un brazo más potente, entonces habrá de emplear motores más potentes que el Futaba S128.

Para el motor necesitamos una fuente de alimentación eléctrica CD de 5 V. Aunque los tres micros que utilizaremos para controlar el brazo poseen fuentes de alimentación de 5 V en la puerta para ampliación o para el usuario, estos voltajes están diseñados para activar circuitos electrónicos externos a niveles TTL. La corriente máxima que se debe tomar de estas fuentes de 5 V es de 100 mA. Dado que un servomotor en carga máxima tomará 200 mA, no es aconsejable alimentar los servos del brazo desde la puerta para el usuario. Por consiguiente, necesitamos una fuente de alimentación externa. Lo ideal sería un transformador CD de 5 V, pero una alternativa menos onerosa es utilizar un conjunto unido de tres pilas de 1,5 V. La salida total de 4,5 V será suficiente para activar el sistema.

Se necesita un trozo de veroboard de 9 agujeros por 20 franjas para montar conectores en los que enchufar los motores, y dos metros de cable plano de 20 vías para conectar el brazo al ordenador. Quienes posean un BBC Micro habrán de comprar un conector IDC de 20 vías, y los que posean un Commodore 64 comprarán un conector marginal de 0,15 pulgadas y 24 vías para conectar el brazo a la puerta para el usuario. El brazo se unirá al Spectrum a través de la interface que hemos construido en *Bricolaje* para controlar el robot. En un próximo capítulo explicaremos en forma detallada las modificaciones necesarias en esta interface.

Para hacer los pivotes del brazo se requieren tres tubos de cobre de 75 mm de largo con un diámetro exterior de 4 mm, junto con tres trozos de tubo de cobre de 75 mm de largo y 5 mm de diámetro. Cada pivote se construirá colocando el tubo de 4 mm dentro del de 5 mm y recortando los extremos a la misma longitud. Por lo tanto, es importante que los tubos que compre se puedan colocar firmemente uno dentro del otro pero con suficiente amplitud como para que el tubo interior pueda girar libremente.

El montaje principal del brazo se une a la caja base mediante un cojinete de bolas. El cojinete debe estar rebordeado y tener un diámetro interno

Lista de componentes

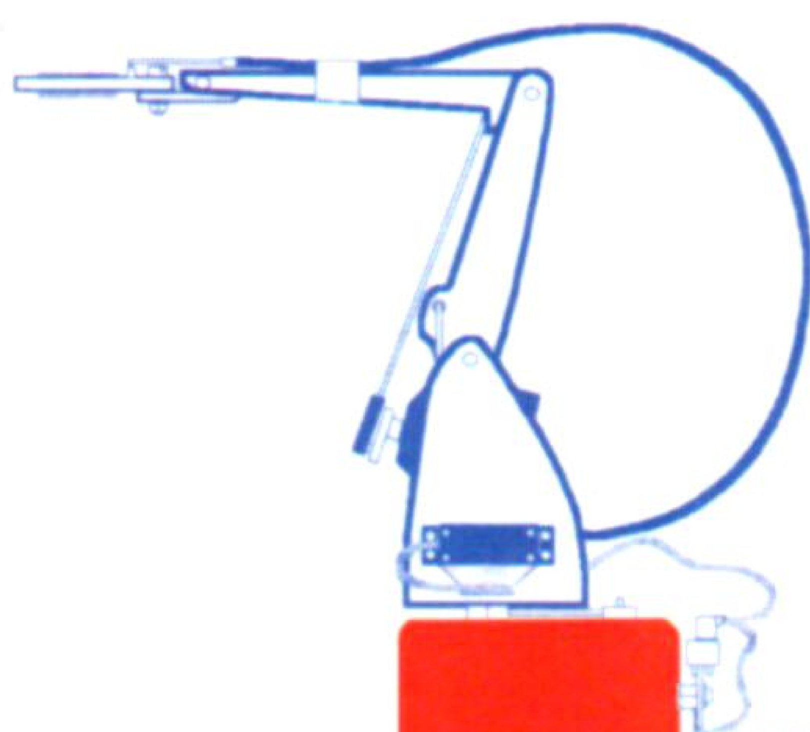
- Algunos tornillos para metales y tuercas 8 BA para montar la pinza
- 5 mm de separador de 6 mm de diámetro interno
- colas epoxy y para madera
- 16 tornillos de 10 mm sin tuerca y arandelas de goma para montar los motores
- 5 cm de espuma para las mandíbulas de la pinza
- Soldadura y cinta aislante

La caja base

TARUGO DE ACERO
PARTE SUPERIOR
HUSILLO DEL MOTOR
SERVOMOTOR
BASE FIJACION DE 12,5 MM
BASE FIJACION DE 6 MM
EXTREMO
EXTREMO
LADO

VISTA INFERIOR DE LA PARTE SUPERIOR DE LA BASE
TARUGO DE ACERO DE 6 MM
BASE FIJACION DE 6 MM
PARTE SUPERIOR DE LA CAJA
BASE FIJACION DE 12,5 MM
SERVOMOTOR

Kevin Jones



de 6 mm. Estos cojinetes a menudo se utilizan para las ruedas delanteras de los coches a escala y se pueden conseguir en tiendas de modelismo. Para alargar el apoyo, también necesitaremos 25 mm de tarugo de acero de 6 mm. Este tarugo se fijará firmemente a la base y se encajará ajustadamente a través del cojinete.

La pinza se opera desde un motor montado más abajo en el ensamblaje del brazo. El motor debe conectarse a la pinza utilizando un enlace flexible de 500 mm de largo (denominado a veces *culebra*). En esencia, éste es un alambre de acero dentro de una manga de plástico. Los aviones a escala con frecuencia emplean este tipo de enlace para los alerones. Se necesitarán 50 mm de alambre duro (el cable de piano es ideal) para abrir y cerrar la pinza, y éste debe atarse al extremo de la culebra con un hilo de fusión a 5 A. Los miembros del brazo se mueven mediante un sistema de varillas. Para ello se utilizarán cuatro varillas de acero de 2 mm de grosor y 150 mm de longitud. Las varillas se pueden conectar a los discos del motor ya sea directamente o bien con pequeños cojinetes de bola.

Construcción de la caja base

Empiece por pegar con cola la base de fijación de 12,5 mm a la parte superior. Luego taladre los agujeros del pivote principal para alojar al tarugo de acero de 6 mm. Taladre atravesando la parte superior y la base de fijación, inserte el tarugo y péguelo bien en su sitio. Pegue las otras dos bases en su lugar, en la parte superior. Estas bases de fijación se utilizan para montar uno de los servomotores. Taladre el otro agujero de la parte superior de modo que a través del mismo pueda deslizarse el husillo, y atornille el motor sobre las bases, utilizando 4 tornillos sin tuerca y arandelas de goma. Pegue los extremos y el lado de la caja. Una vez secos, lije las esquinas y los bordes

Organizar los datos

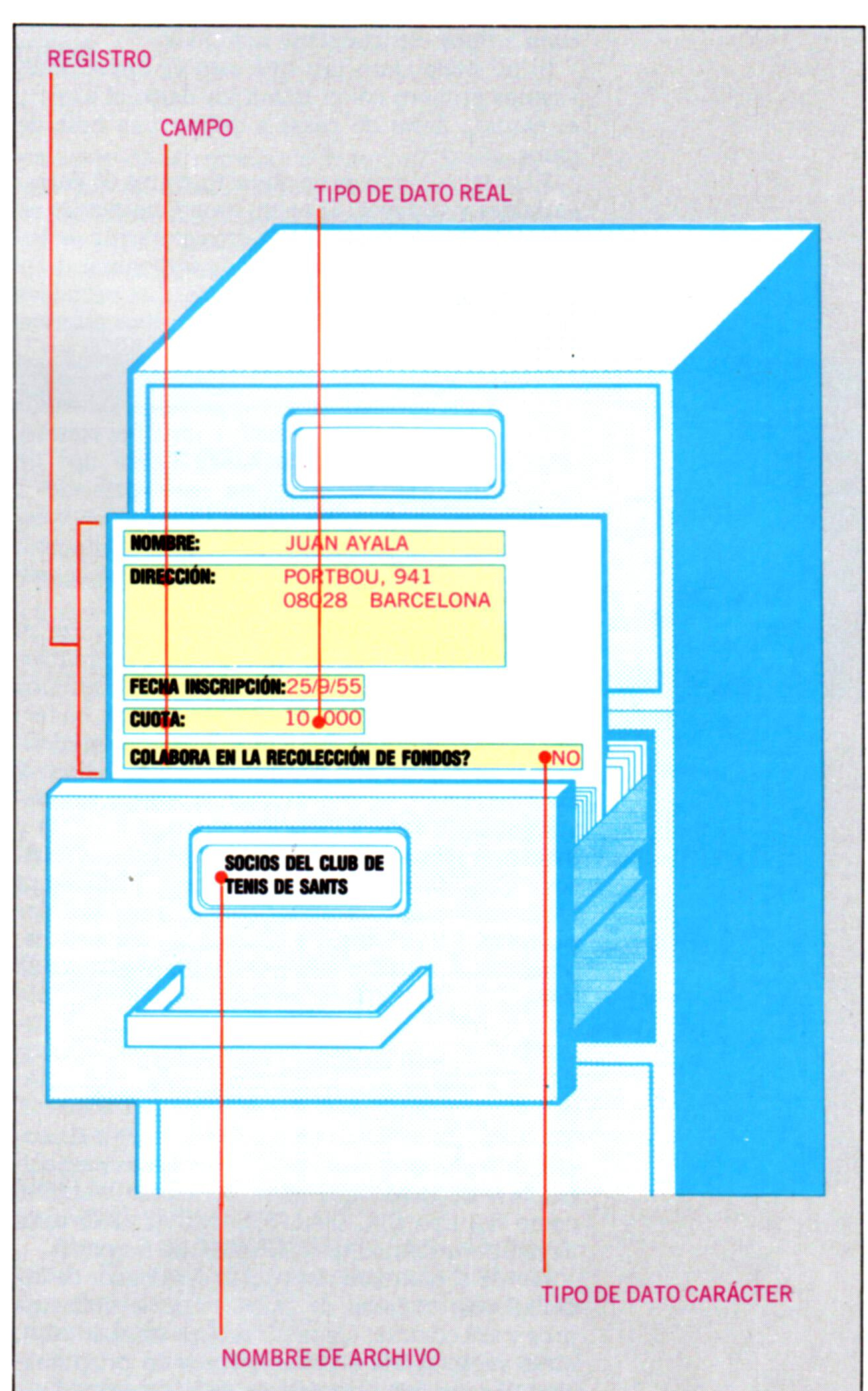
En este primer capítulo de esta nueva serie nos referiremos a las ideas en que debe basarse un sistema eficaz y bien estructurado

Suelen manipularse grandes cantidades de datos estructurados en entidades tan diferentes como pueden ser una secretaría de club, una agencia inmobiliaria o el departamento administrativo de un hospital. Un ordenador y un programa para administración de bases de datos pueden ser de ayuda para mantener organizada y actualizada la información. Pero antes de hablar de cómo se utilizan los programas de bases de datos, es importante pasar revista a los conceptos fundamentales que las mismas implican.

Para el programador de ordenadores, los datos se pueden considerar como estructurados o desestructurados. Los datos desestructurados son unidades aisladas, por lo general almacenadas en variables o constantes. Ejemplos de ello son NUMERO_DE_INTENTOS:=NUMERO_DE_INTENTOS+1 o IF ACIERTO=1 THEN PROCEDIMIENTO AUMENTAR. En estos dos ejemplos, NUMERO_DE_INTENTOS y ACIERTO son variables; en el primer caso, se está cambiando el valor de las variables, mientras que en el segundo se lo está comparando (con una posible bifurcación a un procedimiento). Es evidente que los objetivos NUMERO_DE_INTENTOS y ACIERTO sólo pueden tener un valor en un momento dado.

Los programadores de ordenadores, en especial quienes sólo están familiarizados con el BASIC, están acostumbrados a pensar en los datos de esta forma, como no estructurados; pero en la vida real todos estamos habituados a pensar en datos estructurados y a dar por sentadas dichas estructuras. Piense en sus amigos, por ejemplo. Ciertamente, en este sentido usted tiene su propia "estructura de datos". Conoce sus nombres, sabe su sexo, cuál es su edad (aproximadamente), tal vez sepa en qué trabajan y quizás incluso cuánto ganan, más otros detalles. A cada conjunto de datos relacionados, como éstos que hemos mencionado, lo llamamos *registro*. Cada registro se compone de uno o más campos (que pueden ser de tipos de datos similares o no), y el conjunto de registros se denomina *archivo*.

Para poner un ejemplo concreto, consideremos al secretario de un club de golf que tiene inscritos 500 socios. Para tener todo en orden, el secretario lleva un índice de fichas: una ficha para cada socio. Cada ficha contiene varias clases (tipos) de información (datos) acerca de un solo socio. La información podría incluir: APELLIDO, NOMBRE, SEXO, EDAD, AÑO DE INSCRIPCION, PAGO LA CUOTA?, SUELDO, NUMERO DE SOCIO, etc. En la terminología informática, cada uno de estos elementos de datos dentro del registro se denomina *campo*, y los diversos campos contienen diferentes tipos de datos. APELLIDO y NOMBRE serán, obviamente, series de caracteres, de modo que 137 4662 no sería un dato apropiado para el nombre de un socio. La entrada para SEXO sería un tipo booleano, capaz de tomar sólo uno de dos valores binarios. EDAD será un tipo entero (siempre que no deseemos saber que la edad de un socio es 37,224). AÑO DE INSCRIPCION será un valor entero con una gama limitada. Supondremos que AÑO DE INSCRIPCION = 1066 es inapropiado, aun cuando el archivo contenga registros de antiguos socios. Asimismo, también es impro-



Ian McKinnell

bable AÑO DE INSCRIPCION = 2001, a menos que el reglamento del club permita la inscripción anticipada de niños que se harán socios en el futuro. SUELDO sería un dato de tipo real; alguien podría tener un sueldo de 12 345,67 libras esterlinas.

Los programadores deben estar muy al tanto de los tipos de datos de que disponga el lenguaje de programación escogido. Algunos lenguajes, incluyendo al BASIC y al C, son pobres en cuanto a tipos, mientras que otros, como el PASCAL, insisten en que todos los elementos de datos se adecuen a tipos estrictamente definidos.

Si no tiene claro por qué esto es importante, veamos primero cómo tratan los datos el BASIC y el PASCAL, antes de pasar a definir una base de datos.

El BASIC sólo reconoce dos estructuras de datos: variables y archivos. Si en un programa escrito en BASIC se requiere una mayor estructuración de los datos, el programador habrá de imponerla, porque el lenguaje no la proporciona. Las variables del BASIC pueden ser de numerosos tipos diferentes, incluyendo valores enteros (MARCADOR%=7, p. ej.), reales de precisión simple (SUELDO!=1234.56), reales de doble precisión (SALARIO DE JOHN=123456.66666666666) y variables en serie (MARAVILLOSO$= Mi Computer). Como tipo de constante, en BASIC no existe nada parecido a PI=3.141592; por supuesto, uno siempre puede establecer el valor de la variable PI, pero igualmente puede incluir más adelante en el programa la sentencia PI=6.2.

Otros lenguajes de programación (y quizá el mejor ejemplo conocido sea el PASCAL) proporcionan varios tipos predefinidos y permiten que los programadores definan otros tipos nuevos, en términos de los tipos ya existentes. Los tipos predefinidos del PASCAL son constantes (no variables) y variables de tipo char (caracteres), integer (números enteros), real (números reales como 12.71) y boolean (datos binarios que son o verdaderos o falsos). Conjuntos, matrices, registros y archivos ya están todos incorporados, y el programador puede fácilmente definir nuevos tipos de datos. Veamos ahora cómo se podría crear el tipo de datos DIA.

```
TYPE
  DIA=(LUNES,MARTES,MIERCOLES,
        JUEVES,VIERNES,SABADO,DOMINGO);
```

Toda variable del tipo DIA que se utilice en el programa podrá tener sólo uno de los valores especificados, de modo que si se definiera DIA_LIBRE como del tipo DIA, DIA_LIBRE:=NOVIEMBRE sería ilegal, pero DIA_LIBRE:=DOMINGO se aceptaría.

Desde el punto de vista del programador de ordenadores, el tema de tipos estrictos tiene sus pros y sus contras; algunas veces es un obstáculo, otras veces ayuda a evitar errores de programación. Hemos cubierto algunos de los conceptos relativos a la definición de los tipos de datos, de modo que veamos ahora la importancia de las estructuras de datos en relación con las bases de datos.

¿Qué es una base de datos?

Una base de datos es un conjunto cualquiera de datos estructuralmente relacionados entre sí. Convencionalmente, los datos se organizan como un archivo (con su propio nombre de archivo), el archivo consta de registros (cada uno con su propio número de registro) y los registros están compuestos por uno o más campos de datos, conteniendo los campos uno o más tipos de datos.

Suponiendo que su base de datos está organizada como un índice de fichas de un fichero, la clave básica de cualquier entrada será, casi con toda seguridad, un nombre dispuesto por orden alfabético. Si existe la necesidad de buscar a un socio que se llame Martínez, usted irá pasando las fichas hasta hallar el archivo M, luego irá buscando con más detenimiento los apellidos que empiecen con Ma hasta que finalmente hallará las entradas que haya para Martínez. Sin embargo, si se le pidiera que buscara cuántos socios de sexo femenino tiene en el archivo, encontrar la respuesta sería laborioso. La tarea de hallar los datos llevaría aún más tiempo que si alguien deseara saber cuántos socios hay mayores de 36 años que ganen 11 000 libras al año.

Pero ésta es exactamente la clase de labor que puede hacer un programa de base de datos con sólo pulsar algunas de las teclas del ordenador. Hablando con exactitud, la base de datos es información, y el programa que ayude al usuario a entrar datos y manipularlos será lo que se denomina un *administrador de bases de datos*. Para mantener esta distinción entre el conjunto de datos y el programa que lo manipula, llamaremos al programa *administrador de bases de datos*, o DBM.

Las capacidades de los DBM varían considerablemente. Los más simples son poco más que agendas de direcciones electrónicas capaces de recuperar un registro cuando se les da un parámetro simple, como NOMBRE = ?. Los DBM más sofisticados poseen sus propios lenguajes de programación incorporados que permiten manipular la información de la base de datos de manera sumamente compleja, incluyendo la exportación de campos de datos a otros programas (como software de nómina de pagos, de facturación o de tratamiento de textos). En esta serie analizaremos los DBM, desde el más barato y sencillo hasta el más avanzado, y aprenderemos cómo usarlos de la forma más adecuada. Pero, no obstante, antes de hacerlo haremos una revisión sobre la tipología de los datos. Un buen DBM permitirá que el usuario defina el tipo de datos disponible para cada campo, y rechazará una entrada como NOMBRE =143326 o AÑO DE INSCRIPCION = 1066. Asimismo, dará un mensaje de error si usted intenta hallar registros en función de IF NOT (HOMBRE OR MUJER) AND SALARIO<0. Los buenos administradores de bases de datos, a diferencia de los lenguajes de programación, necesitan poseer una estricta comprobación de tipos para impedir que se entren datos erróneos cuando se están creando o modificando archivos.

En futuros capítulos analizaremos cómo organizan sus registros los diversos DBM. Consideraremos las bases de datos relacionales y de archivos múltiples, los campos y registros de longitud fija en comparación con los de longitud variable, creando bases de datos y extrayendo información de bases de datos. Para ilustrar los conceptos implicados, nos concentraremos en tres DBM: *Archive* de Psion, *dBase II* de Ashton Tate y *Card Box* de Caxton Software, si bien veremos someramente algunos otros paquetes.

Principales eventos en el BBC Micro

Los denominados "eventos" son señales de interrupción a las que la CPU puede permitirse no hacer caso

Hay numerosas situaciones en que las señales "eventos" son generadas en el BBC Micro: así, cuando se entran caracteres en el buffer de entrada, después de la generación de una señal sincrónica vertical, al terminarse una conversión en el convertidor A/D y cuando se pulsa la tecla Escape. Algunas de estas señales son ocurrencias de la máquina que también generan interrupciones, y son tratadas con el vector IRQV1.

Los eventos, pues, nos permiten aprovechar algunas de estas IRQ de máxima prioridad por medio de un método casi de "puerta de atrás". Para que se generen eventos y actúe en consecuencia el ordenador hay que "habilitarlos", o activarlos. Una vez activado, en cuanto sucede el evento particular, la CPU salta a la rutina cuya dirección de inicio se retiene en el vector en la dirección &220 y 221. Para que la rutina detenga su ejecución cuando se ha generado un evento, hay que *desactivar* éste. La activación y desactivación se hace por medio de dos llamadas OSBYTE. El siguiente cuadro muestra los eventos disponibles y las llamadas OSBYTE empleadas para su activación y desactivación. Aquí sólo damos los eventos más importantes y más comúnmente empleados.

Instrucciones para activar/desactivar eventos

Evento	Activar	Desactivar
Buffer de salida vacío	*FX14,0	*FX13,0
Buffer de entrada lleno	*FX14,1	*FX13,1
Ha entrado un car. en buffer entr.	*FX14,2	*FX13,2
Se ha completado conv. A/D	*FX14,3	*FX13,3
Inicio de sincronización vert.	*FX14,4	*FX13,4
Reloj de interv. se ha agotado	*FX14,5	*FX13,5
Se ha pulsado Escape	*FX14,6	*FX13,6

Así, por ejemplo, para activar el evento El reloj de intervalos se ha agotado ejecutaríamos simplemente *FX14,5 desde el BASIC, o su equivalente en lenguaje máquina. Después de activar este evento, la rutina cuya dirección se encuentra en el vector de eventos se ejecutará en cuanto el reloj de intervalos llegue a cero. Por esta razón, es importante que haya en esa dirección un fragmento de código máquina, de lo contrario puede ocurrir un crac.

Sin duda habrá notado que hay un único vector de eventos y, en cambio, toda una gama de eventos diversos. Y es que todos los eventos provocan un salto a la misma rutina, que debe por ello estar escrita de tal modo que les tenga en cuenta a todos ellos. ¿Cómo se determina cuál ha sido el evento causante de la entrada de la rutina? Si sólo se ha activado un evento, la respuesta es obvia; de otro modo hemos de buscar la respuesta en el contenido del registro A cuando se entra en la rutina. El cuadro siguiente muestra el contenido de varios registros a la entrada de la rutina apuntada por el vector de eventos.

Contenido de registros a la entrada de rutinas de eventos

Evento	Registro A	Registro X	Registro Y
Buffer de salida	0	Núm. del buffer	#
Vaciado del buffer de entrada	1	Núm. del buffer	#
Vaciado del carácter en buffer entr.	2	#	Carácter
Convertidor A/D	3	#	Núm. canal
Sincronización vertical	4	#	#
Reloj de intervalos	5	#	#
Condición Escape	6	#	#

En este cuadro, # indica que el contenido de registro no está especificado. Esto quiere decir que es posible, al entrar en la rutina de tratamiento del evento, comprobar el valor que está en el registro A de modo que se pueda determinar el evento causante de la entrada de la rutina.

Cuando usted escribe una rutina de tratamiento de eventos, debe, como primera tarea al entrar en la rutina, guardar el valor del registro. A diferencia de las facilidades de tratamiento de interrupciones en el BBC Micro, el tratamiento de eventos *no* guarda automáticamente el contenido del registro A y del registro de estado en la pila, por lo que tenemos que hacerlo nosotros. El tratamiento de eventos devuelve el control al programa interrumpido con una instrucción RTS. Dejamos aquí la teoría para pasar a examinar los eventos, uno a uno, con varios ejemplos.

- *Evento 0:* Se genera cuando se vacía el buffer de salida. El número del buffer empleado es el mismo que el usado por OSBYTE 21, que ya vimos al referirnos a la llamada OSBYTE.
- *Evento 1:* Se genera cuando se llena el actual buffer de entrada y se intenta introducir un nuevo carácter. El número especificado es el mismo que para OSBYTE 21. El código ASCII que está en el registro Y a la entrada del tratamiento del evento es el código que ya no cabe en el buffer.
- *Evento 2:* Generado siempre que entra un carácter en el buffer de entrada, por lo general después de ser pulsada una tecla. El siguiente programa muestra los elementos básicos de un tratamiento de eventos, generando un VDU 7 al pulsarse una tecla.

Esto es válido incluso cuando se pulsan teclas y el programa en BASIC se está ejecutando.

```
10 DIM mc% (100)
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [ OPT I%
50 PHP
60 PHA
70 TXA:PHA
80 TYA:PHA
90
100 LDA # 7
110 JSR &FFE3
120
130 PLA:TAY
140 PLA:TAX
150 PLA
160 PLP
170 RTS
180
190 ]:NEXT I%
200 ?&220=mc%MOD256:REM byte inferior
    de la dirección colocado en el vector
210 ?&221=mc%DIV256:REM byte superior de la
    dirección
220 *FX 14,2
230 REM activa el evento sólo después de
240 REM establecida la rutina
250 REPEAT
260 PRINT I%
270 I%=I%+1
280 UNTIL FALSE
```

● *Evento 3:* Se genera cuando uno de los cuatro canales del convertidor A/D ha completado una conversión. Este evento tendrá lugar cada cinco o diez milisegundos, según la proporción de conversión escogida para el convertidor.

● *Evento 4:* Se genera al inicio de un impulso de sincronización vertical. Sucede 50 veces por segundo y puede emplearse como fuente de impulsos de reloj.

● *Evento 5:* Es probablemente el más útil de los eventos en el BBC Micro. Además de la variable TIME, el OS del BBC Micro actualiza un *reloj de intervalos* cada centésima de segundo. Ya tuvimos ocasión de analizar este reloj a propósito de ciertas llamadas OSWORD. El reloj es incrementado con regularidad y al llegar a cero se genera un evento. Es claro que esto nos permite interrumpir el proceso normal a intervalos regulares para que realice otra tarea concreta. El reloj de intervalos se escribe a través de la llamada OSWORD con A=4 y se lee por medio de OSWORD con A=3.

Si deseamos que el sistema genere un evento después de transcurrido un segundo, habremos de cargar el reloj con el valor máximo que puede representarse menos 100. Al incrementarse el reloj, cruzará el cero después de 100 tictacs. Dado que es una variable de cinco bytes, su valor máximo antes de tropezar con el cero es &FFFFFFFFFF. He aquí un programa que ilustra este evento, generando periódicamente un pitido:

```
10 DIM mc% (100), delay% 10
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [ OPT I%
50 PHP:PHA
60 TXA:PHA
70 TYA:PHA
80 LDA # 7:JSR&FFE3
90 . clock set / restablece el reloj para la siguiente
   interrupción
100 LDX # delay% MOD 256 / establece el bloque de
    OSWORD
110 LDY#delay%DIV 256
120 LDA#4
130 JSR&FFF1
140 PLA:TAY
150 PLA:TAX
160 PLA:PLP
170 RTS
180 ]:NEXT I%
190 ?&220=mc%MOD256
200 ?&221=mc%DIV256
210 !delay%=&FFFFFF9C:delay%?4=&FF
220 *FX 14,5
230 CALL clock set: REM pone el reloj en marcha
240 END
```

● *Evento 6:* Se genera al pulsar la tecla Escape.

Hay además otros tres eventos a nuestra disposición, y suceden si se detecta un error en el sistema RS423, un error en el Econet o un Evento Especificado por el Usuario. Con esto damos por concluido el repaso de los eventos más útiles empleados por el OS del BBC Micro. El próximo capítulo ofrecerá una visión general de cómo actúa el sistema operativo al completo.

Ejercite sus conocimientos

El siguiente programa muestra algunos de los principios examinados hasta aquí en nuestro curso sobre el OS del BBC Micro. Examine cuidadosamente el listado, entre el programa y ejecútelo. Trate de adivinar: 1) qué operación está realizando el programa; 2) cómo funciona éste. En el próximo capítulo se dará una cumplida explicación de él, junto con otros programas de muestra. Una pista: el nombre de la variable en la línea 80 puede orientarle por el buen camino

```
20 PROCassemble
30 END
50 DEFPROCassemble
70 DIM MC% &FF
80 oldvec=?&20E+256*?&20F
90 FOR pass=0 TO 3 STEP 3
100 P%=MC%
120 [OPT pass
130 . temp EQUB 00
140 . start
150 PHP
160 STA temp
170 TXA:PHA
180 TYA:PHA
200 JSR check
230 PLA:TAY
240 PLA:TAX
250 LDA temp
260 PLP
270 JMP oldvec
290 . check
300 LDA temp
310 CMP#65
320 BMI out
330 CMP #91
340 BPL out
350 ADC #32
360 STA temp
370 .out RTS
390 ]:NEXT
400 ?&20E=start MOD 256
410 ?&20F=start DIV 256
420 ENDPROC
```


R T Y U I O P
D F G H J K L
X C V B N M

10079
9 788485 822836